LATHAM & WATKINS LLP
Douglas E. Lumish (SBN 183863)
doug.lumish@lw.com
Gabriel S. Gross (SBN 254672)
gabe.gross@lw.com
Arman Zahoory (SBN 306421)
arman.zahoory@lw.com
Rachel S. Horn (SBN 335737)
rachel.horn@lw.com
140 Scott Drive
Menlo Park, California 94025
Telephone: (650) 328-4600
Facsimile: (650) 463-2600

Attorneys for Defendant and Counterclaimant Skyryse, Inc.

UNITED STATES DISTRICT COURT

CENTRAL DISTRICT OF CALIFORNIA

MOOG INC.,

Plaintiff,

v

SKYRYSE, INC., ROBERT ALIN PILKINGTON, MISOOK KIM, and DOES NOS. 1-50,

Defendants.

SKYRYSE, INC.,

Counterclaimant,

v

MOOG INC.,

Counterclaim-Defendant.

CASE NO. 2:22-cv-09094-GW-MAR

DECLARATION OF NIKOLAUS BAER IN SUPPORT OF DEFENDANT AND COUNTERCLAIMANT SKYRYSE, INC.'S OPPOSITION TO MOOG'S MOTION TO ENFORCE COMPLIANCE WITH MARCH 11, 2022 STIPULATED TRO AND FOR MONETARY AND ADVERSE INFERENCE SANCTIONS

Judge: Hon. George H. Wu
Crtrm: 9D

REDACTED VERSION OF DOCUMENT PROPOSED TO BE FILED UNDER SEAL

DECLARATION OF NIKOLAUS BAER

I, Nikolaus Baer, declares of follows:

I. INTRODUCTION

1. I am more than eighteen years of age and a citizen of the United States, currently residing in California.

2. I have been retained by counsel for Skyryse, Inc. in connection with the matter *Moog Inc. v. Skyryse, Inc., Robert Alin Pilkington, Misook Kim*, and Does Nos. 1-50, 2:22-cv-09094-GW-MAR. I have been asked to analyze and respond to the Declaration of Kevin Crozier filed on March 16, 2023 ("Crozier Declaration") and the Declaration of Bruce Pixley filed on March 16, 2023 ("Pixley Declaration").

3. I have personal knowledge of the facts and opinions set forth in this Report and, if called to testify as a witness, could and would competently testify to them under oath.

II. SUMMARY

4. Based upon my review of documents produced with the Crozier Declaration and my access to iDS, I have found that Mr. Pixley and Mr. Crozier's methodology provides an insufficient basis to conclude whether a document or its contents actually constitute Moog nonpublic information, notably as I have found several documents and pieces of source code that Mr. Crozier and Mr. Pixley point to as Moog's non-public information are based on information that exists in the public domain and/or information that was developed by Mr. Pilkington prior to his employment with Moog.

5. Mr. Crozier and Mr. Pixley [redacted]. That approach is flawed for multiple reasons, as described further below, including because information that is found on a Moog-issued electronic

device may have originated from outside of Moog and/or may not belong to Moog and also because documents and files that contain language that refers to Moog or purports to identify the contents as confidential or proprietary may nonetheless be publicly available or derived from non-Moog sources.

6. My specific opinions are summarized as follows:

a. Mr. Crozier opines that "Skyryse continues to use the Skyryse Desk Top environment (SDTE) test framework for its software testing activities," that the "SDTE framework is a nearly identical copy of the Moog Desktop Environment (MDTE) test framework that is employed by Moog" (Crozier Decl. ¶ 95), which Mr. Crozier claims constitutes "Evidence of Misappropriation and use of Moog Data after March 11, 2022." (*Id*. at ¶ 49.) Mr. Crozier's opinion is flawed for multiple reasons. First, I have identified source code on Mr. Pilkington's personal laptop, which has been available for review on the iDS platform, that strongly indicates that Mr. Pilkington developed the code underlying MDTE and accompanying RTB spreadsheets containing test information prior to his employment at Moog, as "ASTE." Second, I have also reviewed Skyryse's current source code repository and can confirm that Skyryse has removed the SDTE code Mr. Crozier identifies.

b. Mr. Crozier also opines that his "analysis of SRTOS[1] html files which are eventually compiled to the .chm file (chm file is a compressed html file) shows that it contains numerous identical or slightly modified figures (ie.: SRTOS replaces eRTOS), identical document structure and number word-for-word passages to Moog eRTOS.chm files." (*Id*. at ¶ 103.) Mr. Crozer opines that "[t]his preliminary design document along with source code provided during discovery suggest that the Skyryse SRTOS operating system is copied directly from the Moog eRTOS operating system." (*Id*.) Mr. Crozier claims that these findings also

[1] RTOS refers to Real Time Operating System.

constitute "Evidence of Misappropriation and Use of Moog Data after March 11, 2022." (*Id.* at ¶ 49.) But the idea that Moog and Skyryse's real time operating systems are identical can only be established by reviewing and comparing the object code or source code for the two programs (not design documents), an analysis Mr. Crozier [redacted]. (Crozier Dep. 113:23-114:1; 116:12-14.) I have analyzed the code for both programs and can confirm that they are not identical, which makes sense because a real time operating system by its very nature has to be highly customized and specific to the product to which it applies. In addition, I have also reviewed Skyryse's current code base and confirmed that it no longer contains sRTOS code. Furthermore, eRTOS includes third-party source code that is publicly available and did not originate with Moog. Finally, I have identified an earlier version of the RTOS code on Mr. Pilkington's personal computer that predates his time at Moog. Moog did not identify this pre-existing code and I discovered this code too late to review the full extent of similarities to eRTOS, but its existence further undermines Mr. Crozier's (and Moog's) claims that eRTOS constitutes Moog non-public information.

c. Mr. Crozier also opines that he has identified "Evidence of Use of Moog Non-Public Information in Skyryse Google Drive" and a folder "Discrete IO Slice Package." (Crozier Decl. at ¶100.) Mr. Crozier points to a collection of Python Scripts that "along with software application called Doxygen scan the software source code directories and generate HTML documentation that is used to produce a design document that is required for FAA software certification." (Crozier Decl. at ¶101.) The HTML can get packaged into a compressed HTML file ("CHM"), which is a typical format for help files. Mr. Crozier's opinion is flawed for multiple reasons. First, Doxygen is an open source program for generating HTML documentation from source code. Second, the format of the design documents is defined in publicly available Software Design Description ("SDD") specification. Third, the Python scripts that Mr. Crozier points to includes

third-party source code that is publicly available. Fourth, I have also reviewed Skyryse's current source code repository and can confirm that Skyryse is no longer using the Doxygen Python scripts Mr. Crozier identifies.

d. Mr. Crozier also opines that his review of documents produced in this action reflect to him "that the Moog JIRA document was the basis [of] the Skyryse JIRA document" (*id*. at ¶ 45), and also "that a Moog document, [redacted], became the foundation of the Skyryse document [redacted]." (*Id*. at ¶ 48.) Mr. Crozier also points to an associated SVN guide. (*Id*. at ¶¶25, 27). According to Mr. Crozier, these findings constitute "Evidence of Misappropriation" by Skyryse. (*Id*. at ¶17, 22.) I have reviewed both the SVN and JIRA documents prepared by Skyryse and Moog and can confirm that they are both based on information available in the public domain and well understood by professionals in the field. Indeed, JIRA and SVN are third-party applications, which are neither proprietary nor created by Moog or Skyryse. In addition, my investigation has confirmed that Skyryse is no longer using SVN, which means the SVN guide identified by Mr. Crozier is of no use to Skyryse.

e. Mr. Pixley describes in his declaration that he understands that based on his "analysis of Tri Dao's Moog laptop and the Ivanti log associated with his USB activity," he "found that on February 6, and February 9, 2022, [Mr. Dao] copied 39,278 files to an external USB drive," and that "[a]pproximately one week later on February 15, 2021, Tri Dao plugged the same external USB drive into his Skyryse laptop (IDS S0022) and copied 7,679 files (of the 39,279 file) he originally copied from his Moog laptop to his Skyryse laptop." (Pixley Decl. at ¶¶ 20-21.) Mr. Pixley provides no opinion regarding the files that were allegedly transferred by Mr. Dao to his Skyryse laptop, although I understand that he had access to those files. (iDS container MOOOG-04208.S0022.L01). I have reviewed the files that Mr. Dao allegedly transferred to his Skyryse laptop and can confirm that these files relate to "Adruino," which is an open-source hardware and software

company, project, and user community that designs and manufactures single-board microcontrollers and microcontroller kits for building digital devices.

III. BACKGROUND AND EXPERIENCE

A. Experience

7. I began my formal academic study of computer technology at the University of California, Santa Barbara ("UCSB"), which I attended on a Regents Scholarship. In 2004, I received a Bachelor of Science degree in Computer Engineering from UCSB, with high honors, along with a certificate in Technology Entrepreneurship from UCSB.

8. Professionally, I have developed software for military terrain databases, marine research, optical testing equipment, mobile applications, and medical devices. I have worked at a variety of technology firms as a developer of software, firmware, and drivers. This work has involved developing software in the C and C++ programming languages and in the LabVIEW engineering software from National Instruments.

9. I am a member of the Institute of Electrical and Electronics Engineers ("IEEE") and an officer of the Northern California Scholarship Foundation's Alumni Association.

10. Since 2006, I have consulted on a wide range of matters involving software intellectual property. I have served as both a consulting and testifying expert in over sixty litigation matters as well as serving as an outside analyst for internal technical investigations. These representations all involved software and technical systems. These matters have included assertions of copyright infringement, trade secret misappropriation, patent infringement, breach of contract, software plagiarism, and other misappropriation of intellectual property rights. I have served as a testifying witness in depositions, hearings, and arbitrations. My clients have included Salesforce.com, Nest Labs, Inc., Electronic Arts, Oracle, Applied Materials, and Facebook. In addition to these matters, I have

also served as a Special Master for U.S. District Court, Western District of Tennessee for the matter of *ECIMOS, LLC v. Carrier Corporation* (2:15-CV-2726-JPM-cgc).

11. In providing such services, I have performed source code analyses and I have developed significant expertise in the analysis of software. I am experienced in the art of analyzing the functionality, strengths, and weaknesses of software through the use of detailed source code reviews, debugging, reverse-engineering, utilizing tools for mapping the workflow of the source code, and examining records of the development history. I am skilled in the use of state-of-the-art forensic software to conduct such source code analysis, including conducting comparisons between source code for distinct software products to identify overlapping code, architecture, features, and functions. This includes analyzing software works for overlapping code and offering possible reasons for why such similarities may exist.

12. In connection with my litigation consulting services, I have developed tools and scripts for analyzing software, as well as skill in analyzing claims of copying between and among software works. I have extensive experience in assessing the strengths and weaknesses of claims of intellectual property infringement in the context of software, particularly with respect to claims of copyright infringement, trade secret misappropriation and patent infringement. I also have authored several papers on engineering and software analysis based upon my experience analyzing software, including the article "What, Exactly, Is Software Trade Secret Theft?"[2]

13. A copy of my resume is attached as Appendix A, together with a list of software IP litigation matters that I have been involved with, including cases that I have testified as an expert at trial or deposition.

---

[2] Baer, N. and Zeidman, B., "What, Exactly, Is Software Trade Secret Theft?" Intellectual Property Today, March 2008.

14. My services are being compensated at my standard rate. My compensation is in no way contingent upon or related to my findings and conclusions.

B. Technical Background

15. This section provides brief explanations of some of the technical terminology discussed herein.

1. Source Code

16. Software is developed as source code. Source code is composed of sequences of instructions that cause a computer to perform some functionality. Human developers write and edit source code to provide specific features and functions, so it is generally considered to be human-readable.

17. Source code can be written in a variety of software languages. These languages generally include statements that either define operations or data. Traditionally these statements are compiled, or translated, into another format (machine code), which is essentially a series of ones and zeroes that control the operation of a computer.

18. Source code may include comments: non-functional statements that often are used to document the source code. Comments are not compiled and do not affect the operation of the software program. Although comments generally are not included in the compiled software, they can provide important insight into the functionality and history of a work of source code, and aid in the overall accessibility of the surrounding source code.

19. The functionality of software can be understood at multiple levels of abstraction, where each level is an expression of the information from the level above. These levels can be defined as the overall function, the architecture, the algorithms, and the actual expression of the source code.

20. The overall function level represents the purpose of the software as a whole. This level represents the reason that this software exists, in terms of the problem it addresses and the solution it provides.

21. The architecture level represents the organization of the software system. This includes the arrangement and connections between components of the software system that define how they will operate together to perform the purpose of the software as a whole. This may also include an arrangement or definition of what algorithms will be needed.

22. The algorithm level includes the formulas and patterns necessary for the software components to operate. They provide a particular functionality. In addition to developing novel algorithms, developer often rely upon common algorithms, which generally are known and used to provide some common functionality, such as the reading and writing of files.

23. Software algorithms are implemented in source code. Algorithms often are organized into software elements called methods, functions, routines, and software classes or objects within the source code. Depending on the precision with which an algorithm is defined, a developer may implement that algorithm in source code without a greater understanding of the software system, or even the overall function of the system in which it is incorporated. The specific implementation of an algorithm in source code often reflects the overall function of the software or the purpose of the algorithm, such that identifier names defined by the developer are often related to the purpose of the software and algorithm.

24. An algorithm is not limited to a single implementation. That is, there may be multiple ways to write source code to accomplish the same goal or function, and different developers can create different implementations of the same algorithm because of differences in software languages, identifier names, commenting, order of operations, or even styles. These differences can give a specific source code implementation of a given algorithm a unique identity or

"fingerprint." When comparing source code that overlaps in either the exact expression (that is, they are the same word for word or line for line) or functionality (that is, they accomplish the same goal), these differences, or lack thereof, can reveal whether a given source code work was developed independently or copied from another work.

25. There are numerous algorithms and bodies of source code that are publicly known and available in the public domain. For example, main.c is a generic filename for the main() function universally used for programs written in the C programming language. The evolution of software has included the sharing of algorithms in both papers and books as well as shared research and open-source software. A well-known example of such sharing can be found in the open-source Linux software source code, which can be retrieved, studied, and adapted in accordance with its royalty-free license.[3] Even before the advent of the Internet as we know it, programmers shared source code and academic institutions fostered collaboration around software development.[4] Now, websites like GitHub store both confidential proprietary source code as well as millions of publicly accessible open source projects.[5]

2. Real-Time Operating Systems

26. A computer operating system is a special type of software that manages the operation of, and interactions with, hardware and software resources of a computer. As part of managing the operations of the computer system an operating system must schedule the tasks of the computer system. A real-time operating system ("RTOS") is a special category of operating systems that schedules task to guarantee responsiveness of the system within specific, real-time,

---

[3] "Linux Kernel Licensing Rules." The Linux Kernel. Web. April 24,2023. https://www.kernel.org/doc/html/v4.18/process/license-rules.html

[4] E. Hippel and G. Krogh. "Open Source Software and the 'Private-Collective' Innovation Model: Issues for Organization Science." Organization Science (2003) 14 (2)208-223.

[5] "The Largest Open Source Community in the World" GitHub. Web. April 24, 2023. https://github.com/open-source.

constraints. All processing tasks must be finished within an assigned timeframe and can be switch based on events and priorities. The adherence to time-constraints provided by an RTOS is important for computer systems used in critical operations with real-world and real-time safety considerations. As such, they are often tuned, or configured, to their particular application; that is the needs of one application or one environment may differ from another, so the particular design and implementation of the respective RTOSes also differ and the software, including RTOS, cannot generally be copied from one system to another. There are many popular RTOSes available, such as MQX, VxWorks, Azure RTOS, and the open-source FreeRTOS that are often used as the basis for configuring an RTOS and embedded system.

3. JIRA

27. JIRA is a software development management and issue tracking tool used by organizations to collaborate and plan software development more efficiently. JIRA is produced by the company Atlassian. JIRA provides a system of dashboards, charts, and workflows where various personnel can report issues, assign development tasks, and record and track progress on the various tasks. The personnel are assigned roles for their work in the organization as well as on specific projects. The organization of roles, issues, and tasks and the ability to visualize all the pieces of work required helps organizations operate more efficiently. To my knowledge, JIRA is not proprietary to Moog.

4. Subversion

28. Subversion, commonly referred to as "SVN" us a well-known versions control system, which allows an organization to collaborate and track the development of, for example, source code. SVN is often used as a source code repository, where source code developers working on a project share access to a SVN source code repository for storing their work on a project. The developers retrieve the current versions of source code, check-out, and then record and share

their modification, check-in or commit. When developers commit their changes, they can also input comments and tracking information as part of maintaining a development history and informing other developers of updates. The SVN system tracks the changes to source code files and lines as well as meta data such as the date and time of changes and the user's comments. To my knowledge, SVN is not proprietary to Moog.

5. Doxygen

29. The Doxygen tool is open source software described as "the de facto standard tool for generating documentation from annotated C++ source, but it also supports other popular programming languages."[6] Doxygen generates the documentation in the HTML format, for use in browser or a compressed HTML help files. Doxygen uses configuration and template files to specify how different comments, or comment blocks, left by a developer will be processed into the documentation. The use of comment blocks allows developers to leave more structured, or formatted, text in the source code for inclusion in the resulting documentation. To my knowledge, Doxygen is not proprietary to Moog.

IV. SCOPE

30. I have reviewed the Crozier Declaration, the Pixley Declaration, and the associated exhibits and documents. I have also been provided access to the secure review platform from the third party discovery vendor iDiscovery Solutions ("iDS").

V. ANALYSIS

31. I have performed an analysis of the Crozier Declaration, the Pixley Declaration, and the associated exhibits and documents and select contents of images produced through the iDS review platform as described in more detail below. The assertions by Moog Inc. and declarations of Messrs. Crozier and Pixley

---

[6] "Doxygen" Dimitri van Heesch. WEB https://www.doxygen.nl/index.html. Accessed on April 11, 2023.

are generally vague, and I therefore reserve the right to supplement or amend my opinion, following the production of additional materials and further analysis of the current or additional materials regarding the alleged, and thus far unsupported, claims by Moog, Mr. Crozier and Mr. Pixley regarding Skyryse's alleged use of Moog non-public information.

32. For the purposes of my analysis, I understand that Mr. Pilkington was hired by Moog on July 30, 2012 (Complaint, ECF No. 1, ¶ 12), and I am not aware of any evidence suggesting Moog owns the work he performed or software that he may have developed prior to his employment at Moog.

A. JIRA Guide

33. Mr. Crozier opines that his review of documents produced in this action reflect to him "that the Moog JIRA document was the basis the Skyryse JIRA document" (*id*. at ¶ 45), and also "that a Moog document, [redacted], became the foundation of the Skyryse document [redacted]." (*Id*. at ¶ 48.) According to Mr. Crozier, these findings constitute "Evidence of Misappropriation" by Skyryse. (*Id*. at ¶ 17, 22.)

34. Specifically, Paragraphs 48-57 of Mr. Crozier's Declaration discuss the documents: Moog document [redacted],[7] Skyryse document [redacted] (BIRD_SR_00001465),[8] and the [redacted] (BIRD_SR_00024768).[9] Mr. Crozier asserts that these documents are nearly identical in structure and in various passages, but he performs no analysis and provides no opinion regarding the substance of the documents. In particular, he does not identify anything he contends is Moog non-public information in those documents, let alone identify where any Moog non-public information may be in

[7] Found at 2023.03.16 [400-6] [UNREDACTED] Exhibit A-10
[8] Found at 2023.03.16 [400-6] [UNREDACTED] Exhibit D-2
[9] Found at 2023.03.16 [400-6] [UNREDACTED] Exhibit D-1

Skyryse's documents. Instead, as explained below, the content in those documents includes information that is generally known and even in several instances identical to information in the public domain. Nor does Mr. Crozier provide any opinion that the structure of the documents is purportedly Moog non-public information. Since neither Moog nor its experts provide sufficient information and analysis to distinguish these documents, or the alleged similarities, from information that is available in the public domain, I am not aware of a factual basis for Mr. Crozier's conclusion that they are purportedly Moog non-public information.

35. Mr. Crozier provides several examples as alleged support for his vague assertion that the documents are nearly identical. First, Mr. Crozier shows the table of contents from the Moog document, [redacted], and the table of contents from the Skyryse document, [redacted], (BIRD_SR_00001465), which are reproduced below are similar. However, Mr. Crozier fails to recognize that the tables of contents for both documents are similar to tables of contents for documents available in the public domain, and Moog's table of contents does not contain any content that would provide it anything of value or advantage over the publicly available tables of contents, which are functionally identical.

36. For example, a table of contents that contains information similar to the table of contents in Moog's alleged document can be found within a publicly available document "JIRA 6 Documentation," created by and distributed by Atlassian, which is the supplier of Jira.[10] It can also be found in an earlier "JIRA Administrators Guide," which is also from Atlassian[11], as shown below. Although the exact formatting and arrangement of the documents are different, the same

[10] "Jira 6 Documentation." Atlassian. WEB https://www.cwiki.us/display/JIRA064/JIRA+6+Documentation. Accessed on April 8, 2023. (Ex. B1)

[11] "Jira Administrator's Guide." Atlassian. WEB https://www.oasis-open.org/committees/download.php/51095/jira-manual-config.pdf. Accessed on April 8, 2023. (Ex. B2)

information is contained in both, including for example, information regarding how to configure security, manage global and project permissions, and manage project roles in Jira.

Figure 1 – [redacted] (Moog)

Figure 2 - [redacted] (BIRD_SR_00001465) (Skyryse)

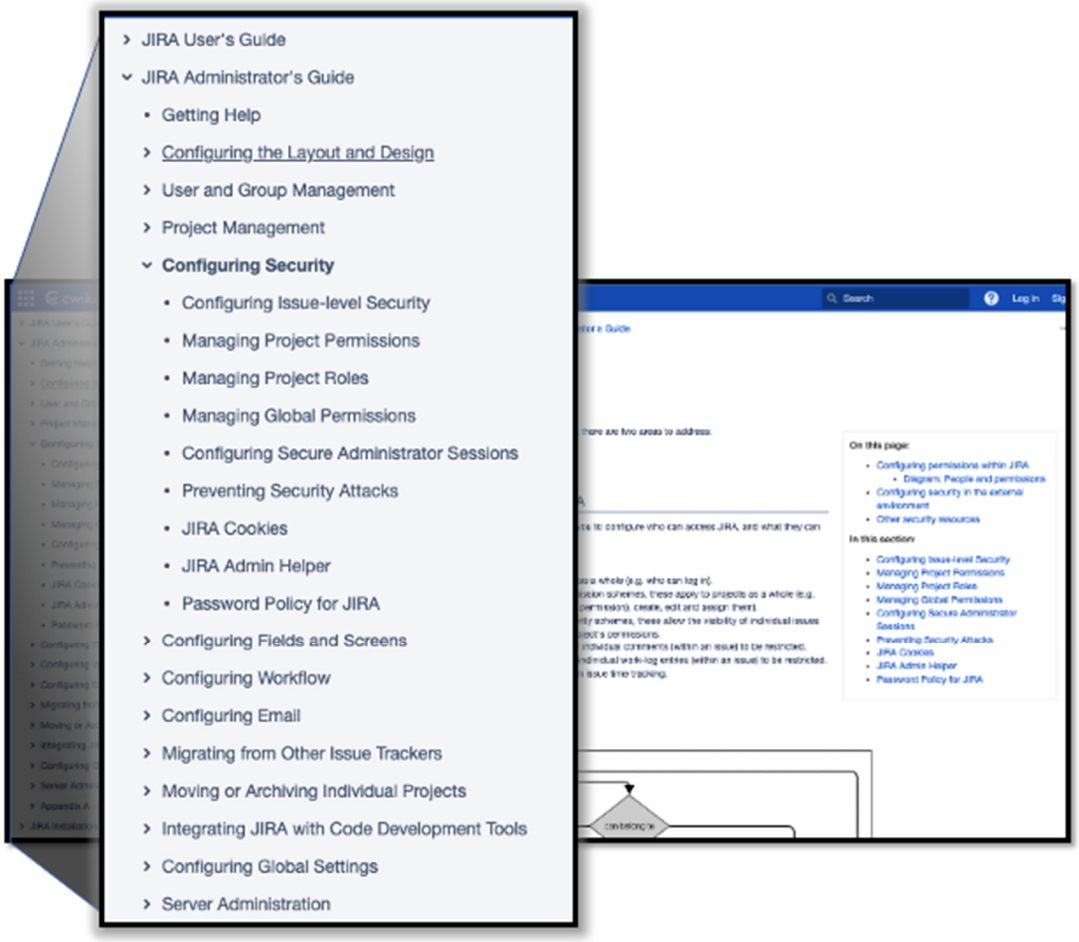


Figure 3 – Publicly Available Table of Contents from JIRA 6 Documentation

1.1. JIRA Administrator's Guide

This manual contains information on administering your JIRA system:

- Look and Feel
- Email
- User Management
- Security
- Project Management
- Configuring Fields and Screens
- Configuring Workflow
- Importing from Other Systems
- Moving or Archiving Individual Projects
- Integrating with a Revision Control System
- Configuration Options & Settings
- Server Administration
- Appendix A - Extending JIRA

Figure 4. Publicly Available Table of Contents from JIRA Administrators Guide

37. The similarities between the Moog document, [redacted], and either the "JIRA 6 Documentation" or the "JIRA Administrators Guide" are striking and become more apparent when some of the details in both documents are examined. For example, when the [redacted] is compared to the section "Configuring Security" of the publicly available document, "JIRA 6 Documentation," or the section "Security " in the public document, "JIRA Administrators Guide," it is clear that Moog's document, [redacted], incorporates this publicly available information, ***and even the identical text***, as shown below. This shows that Moog's document is not just based on publicly available information, the Moog document includes verbatim copies of the publicly available information. Mr. Crozier and Mr. Pixley's failure to conduct this simple public domain search reflects the significant infirmities underlying their expert opinions regarding Skyryse's use of alleged Moog non-public information, including as further described below.

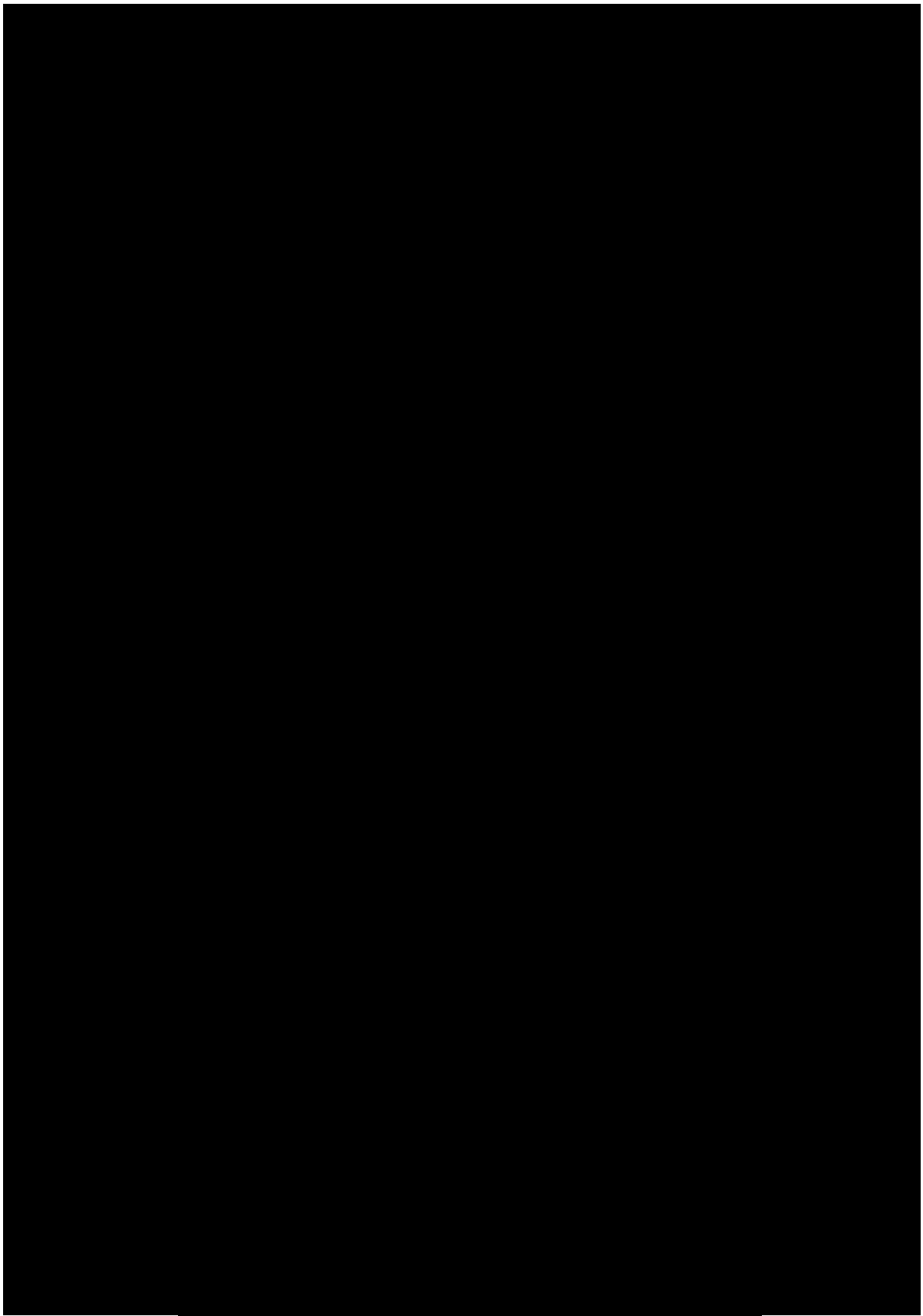
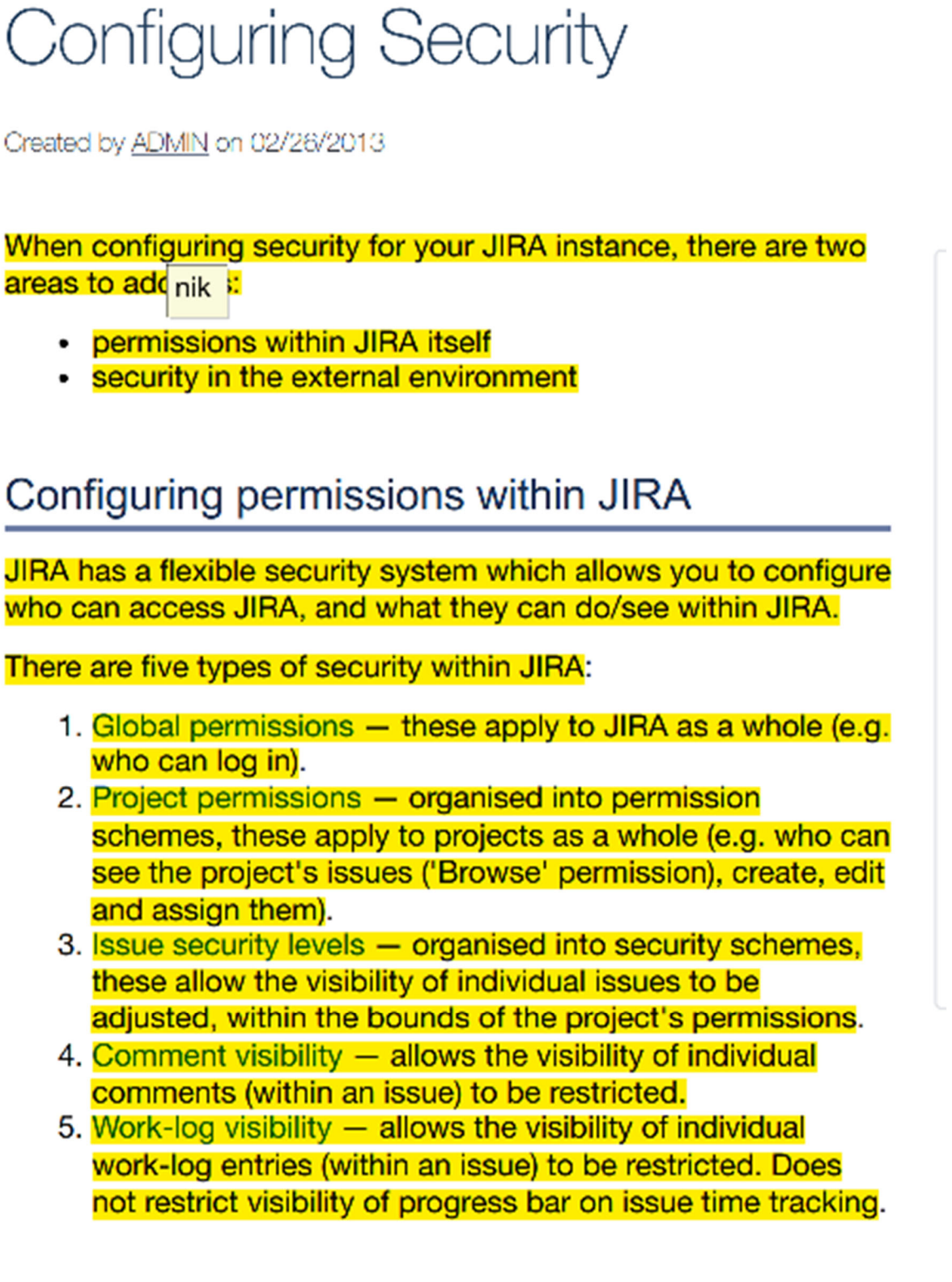


Figure 5 – [redacted] (Moog)



Configuring Security

Created by ADMIN on 02/26/2013

When configuring security for your JIRA instance, there are two areas to adc nik :

- permissions within JIRA itself
- security in the external environment

Configuring permissions within JIRA

JIRA has a flexible security system which allows you to configure who can access JIRA, and what they can do/see within JIRA.

There are five types of security within JIRA:

1. Global permissions — these apply to JIRA as a whole (e.g. who can log in).
2. Project permissions — organised into permission schemes, these apply to projects as a whole (e.g. who can see the project's issues ('Browse' permission), create, edit and assign them).
3. Issue security levels — organised into security schemes, these allow the visibility of individual issues to be adjusted, within the bounds of the project's permissions.
4. Comment visibility — allows the visibility of individual comments (within an issue) to be restricted.
5. Work-log visibility — allows the visibility of individual work-log entries (within an issue) to be restricted. Does not restrict visibility of progress bar on issue time tracking.

Figure 6 – JIRA 6 Documentation – Configuring Security (Publicly Available)

1.6. Security

1.6.1. Security overview

When configuring security for your JIRA instance, there are two areas to address:

- security within JIRA itself
- security in the external environment

1.6.1.1. Configuring security within JIRA

JIRA has a flexible security system which allows you to configure who can access JIRA, and what they can do/see within JIRA.

There are five types of security within JIRA:

1. Global permissions — these apply to JIRA as a whole (eg. who can log in).
2. Project permissions — organised into permission schemes, these apply to projects as a whole. (Eg. who can see the project's issues ('Browse' permission), create, edit and assign them.)
3. Issue security levels *(Enterprise Edition only)* organised into security schemes, these allow the visibility of individual issues to be adjusted, within the bounds of the project's permissions.

Page 64

JIRA Administrator's Guide

4. Comment visibility allows the visibility of individual comments (within an issue) to be restricted.
5. Work-log visibility — allows the visibility of individual work-log entries (within an issue) to be restricted.

Figure 7 – JIRA Administrators Guide (Publicly Available)

38. The Moog document, [redacted] also contains additional examples of publicly available content. For example, [redacted] includes the same content as the page "Managing project roles" on Atlassian's website,[12] as shown below. Despite, this Mr. Crozier opines that this publicly available material is Moog non-public information and allegedly was misappropriated by Skyryse and its personnel.

[12] "Managing project roles". Atlassian. WEB https://confluence.atlassian.com/adminjiraserver/managing-project-roles-938847166.html. Accessed on April 8, 2023. (Ex. B3)

Figure 8 – [redacted] (Moog)

Managing project roles

Project roles are a flexible way to associate users and/or groups with particular projects. Project roles also allow for delegated administration:

- Jira administrators define project roles — that is, all projects have the same project roles available to them.
- Project administrators assign members to project roles specifically for their project(s).
  A project administrator is someone who has the project-specific 'Administer Project' permission, but not necessarily the global 'Jira Administrator' permission.

Project roles can be used in:

- permission schemes
- email notification schemes
- issue security levels
- comment visibility
- workflow conditions

Project roles can also be given access to:

- issue filters
- dashboards

Project roles are somewhat similar to groups, the main difference being that group membership is global whereas project role membership is project-specific. Additionally, group membership can only be altered by Jira administrators, whereas project role membership can be altered by project administrators. Every project has a project lead and every project component has a component lead. These individual roles can be used in schemes, issues and workflows, just like project roles. You assign project/component leads when defining projects or managing components respectively.

Figure 9 – Managing Project Roles from Atlassian (Publicly Available)

39. Analyzing Mr. Crozier's other examples shows the same connections to publicly available information. For example, [redacted] and discussed in paragraphs 51 and 52 of the Crozier Declaration, is an example of dividing the review of the issues and the tasks that are completed to address the issue by multiple teams of an organization. JIRA, provided by Atlassian, facilitates this division of review and approval across multiple roles, as seen by the multiple roles involved in an issue view of a customer request,[13] and the segregation of duties so that multiple people must review and approve work,[14] shown below. Mr. Crozier does not provide any opinion that the roles identified in this document are in any way distinguishable from public information or that the underlying concept of subject matter experts reviewing the subject for which they are an expert is something that is not already well-known to the public. Therefore, this section of [redacted], which is the very portion of the document Mr. Crozier selected as an example of Moog's non-public information is publicly available, and generally known information.

[13] "See everyone involved in a request". Atlassian. WEB https://support.atlassian.com/jira-service-management-cloud/docs/see-everyone-involved-in-a-request/. Accessed on April 8, 2023. (Ex. B4)

[14] "Jira Compliance, Part 1: Approvals and Segregation of Duties" CPrime, Inc. WEB. https://www.cprime.com/resources/blog/jira-compliance-part-1-approvals-and-segregation-of-duties. Accessed on April 22, 2023. (Ex. B5)

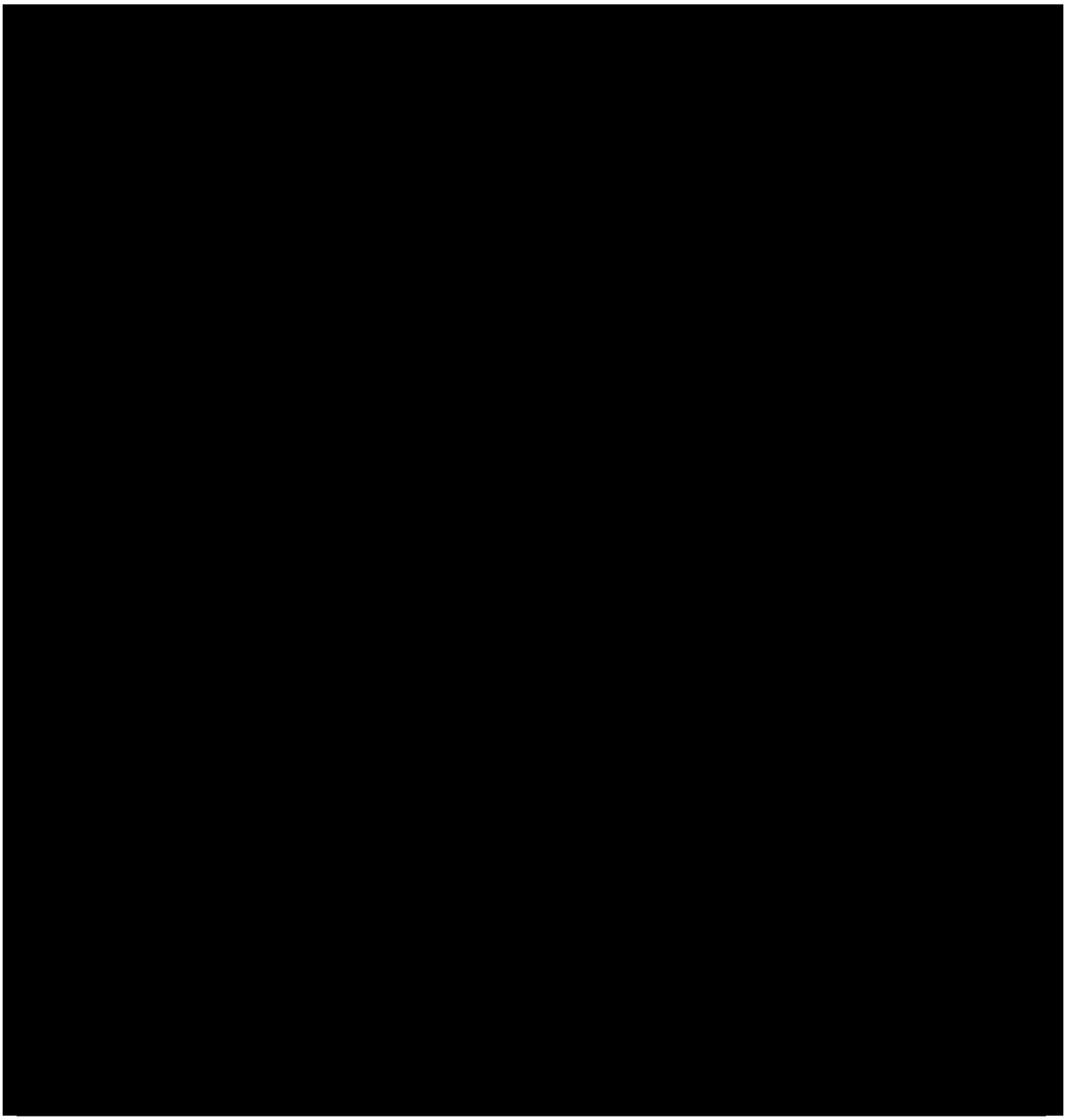
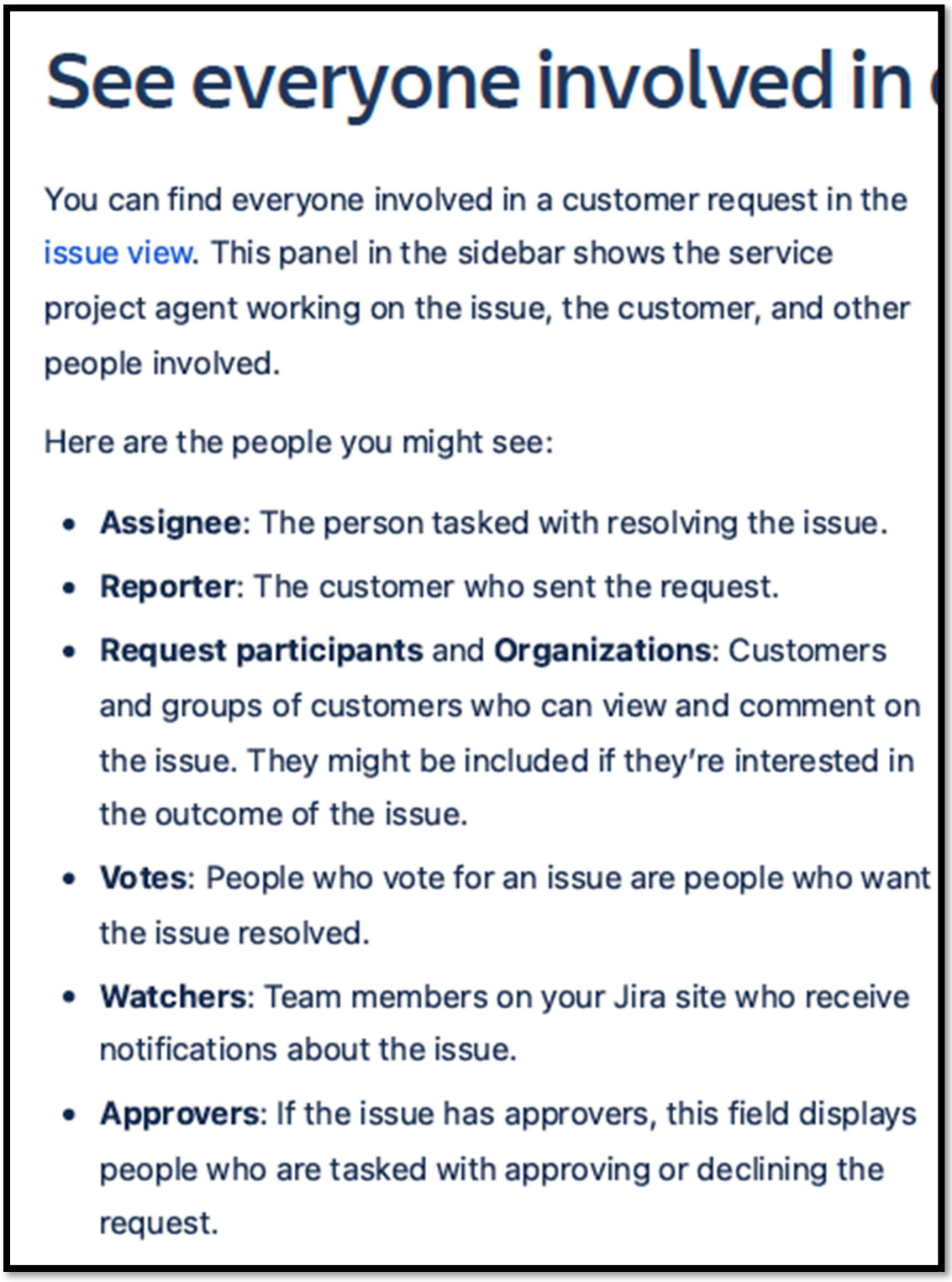


Figure 10 – [redacted] (Moog)



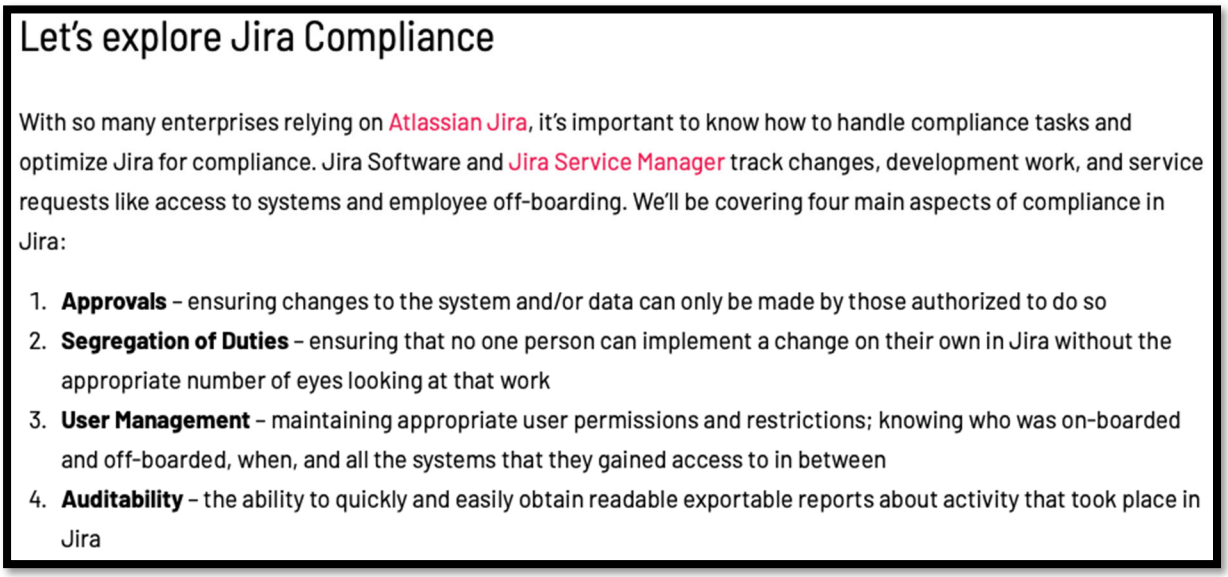
See everyone involved in

You can find everyone involved in a customer request in the issue view. This panel in the sidebar shows the service project agent working on the issue, the customer, and other people involved.

Here are the people you might see:

- **Assignee**: The person tasked with resolving the issue.
- **Reporter**: The customer who sent the request.
- **Request participants** and **Organizations**: Customers and groups of customers who can view and comment on the issue. They might be included if they're interested in the outcome of the issue.
- **Votes**: People who vote for an issue are people who want the issue resolved.
- **Watchers**: Team members on your Jira site who receive notifications about the issue.
- **Approvers**: If the issue has approvers, this field displays people who are tasked with approving or declining the request.

Figure 11 – Managing Project Roles (Publicly Available)



Let's explore Jira Compliance

With so many enterprises relying on Atlassian Jira, it's important to know how to handle compliance tasks and optimize Jira for compliance. Jira Software and Jira Service Manager track changes, development work, and service requests like access to systems and employee off-boarding. We'll be covering four main aspects of compliance in Jira:

1. **Approvals** – ensuring changes to the system and/or data can only be made by those authorized to do so
2. **Segregation of Duties** – ensuring that no one person can implement a change on their own in Jira without the appropriate number of eyes looking at that work
3. **User Management** – maintaining appropriate user permissions and restrictions; knowing who was on-boarded and off-boarded, when, and all the systems that they gained access to in between
4. **Auditability** – the ability to quickly and easily obtain readable exportable reports about activity that took place in Jira

Figure 12 – Managing Project Roles (Publicly Available)

40. The workflow in Moog's [redacted], which Mr. Crozier uses as an example in paragraphs 53 and 54 of his declaration are similar to the Jira Service Desk default problem workflow,[15] as shown below. JIRA is designed with the concept of workflows, where the steps of processing an issue from report to approval of the solution can be designed to guide the organization and its use of JIRA in working through issues. These workflows are publicly known and taught in JIRA documentation, as shown below. Specifically, elements such as processing a "deferred" status are also found within discussions of JIRA Workflows in the Jira Atlassian community. The example workflows, that are publicly known, can include the same basic level of complexity used in the Moog example or involve even more steps and complexity than the Moog example.[16] Therefore, this section of Moog's [redacted]

[15] "Problem management" Atlassian. WEB https://confluence.atlassian.com/servicedeskcloud/managing-problems-with-your-it-service-desk-817562149.html. Accessed on April 8, 2023. (Ex. B6)

[16] "issues in jira that are moved to differed state as "resolved". Due to this, all issues are moving" WEB https://community.atlassian.com/t5/Team-managed-projects-questions/issues-in-jira-that-are-moved-to-differed-state-as-resolved-Due/qaq-p/1008327. Accessed on April 8, 2023. (Ex. B7)

[redacted] that Mr. Crozier also selected as an example of purported Moog non-public information is again publicly available information.

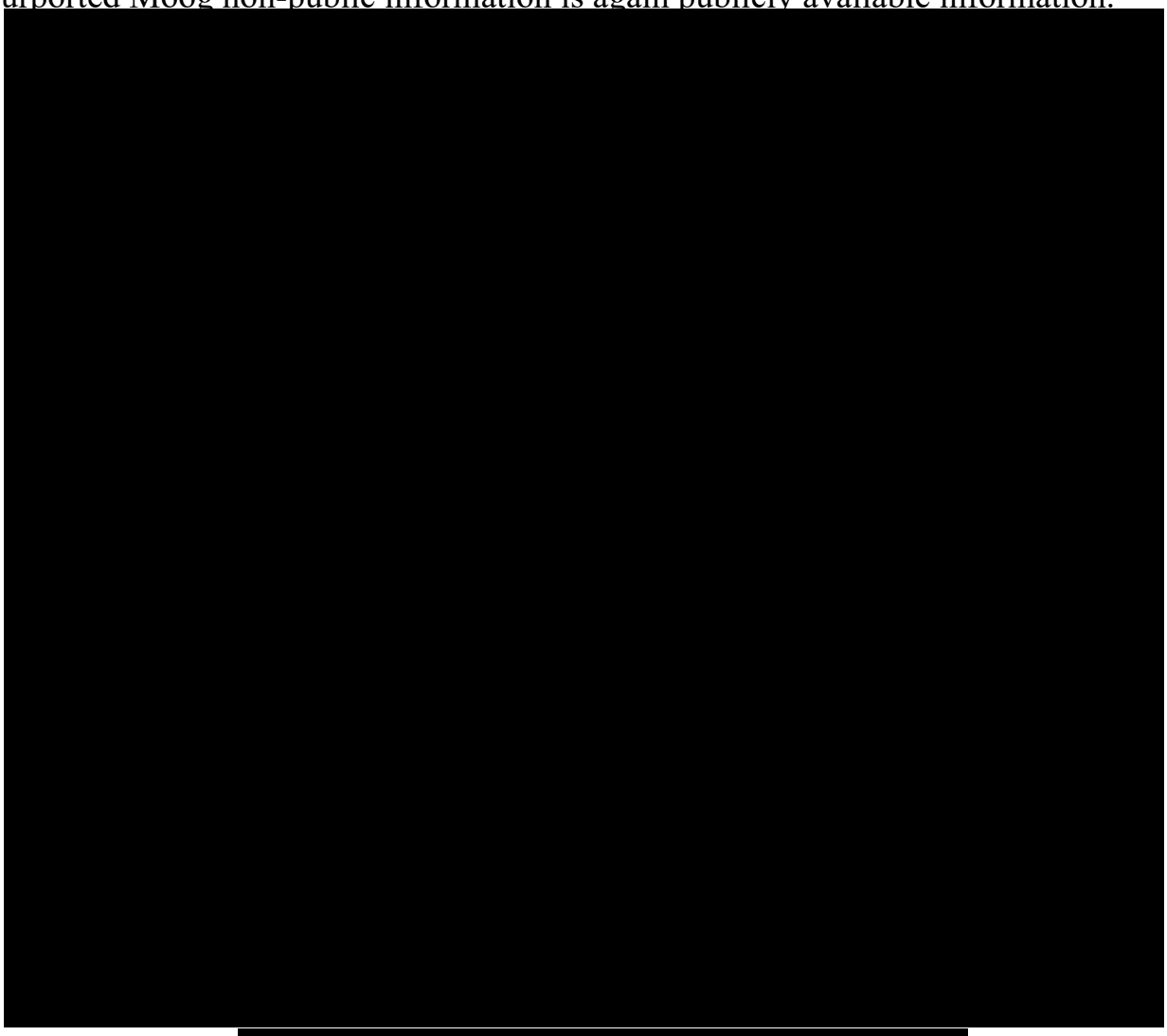


Figure 13 – [redacted] (Moog)

Jira Service Desk's default problem workflow

Your service desk agents can create an issue using the **Problem** issue type. This puts the problem record into the recommended problem workflow.

The workflow follows the basic process above. You can customize it to adapt to the needs of your business.

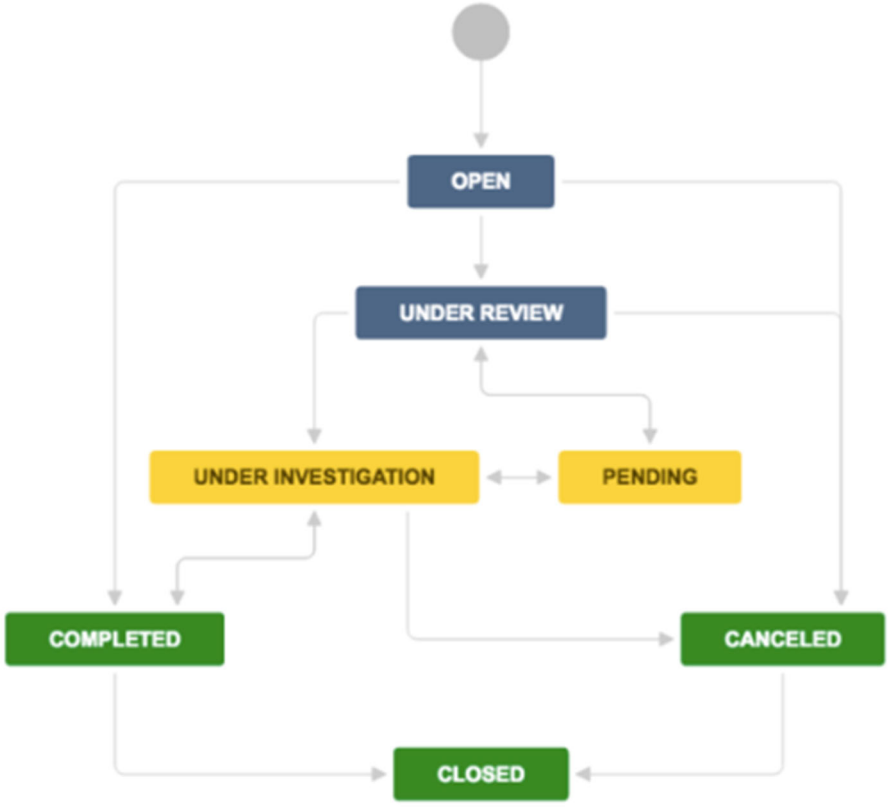


Figure 14 – Jira Problem Management (Publicly Available)

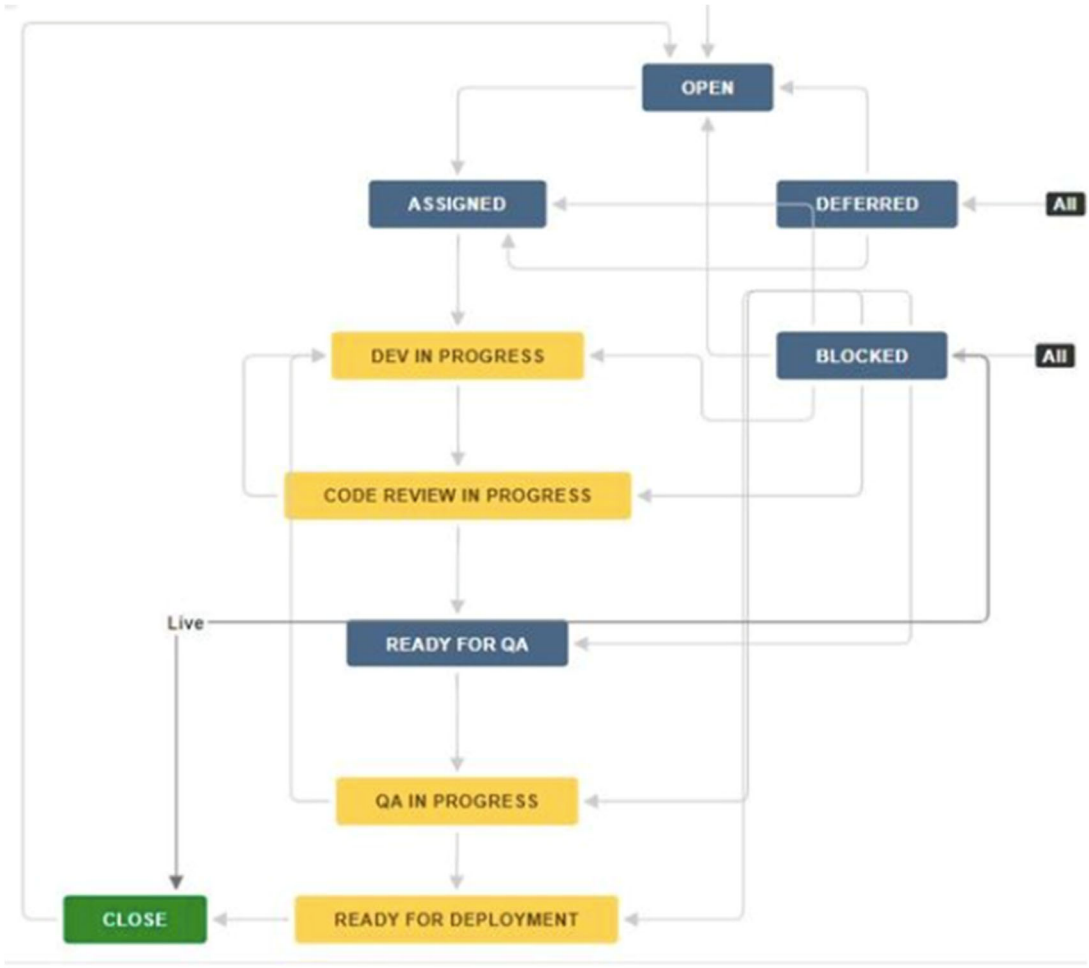


Figure 15 – Jira Community (Publicly Available)

41. Other diagrams in [redacted], which Mr. Crozier also selected as alleged examples of Moog's non-public information, are found identically online and are publicly available. For example, the diagram on people and permissions shown in paragraphs 55 and 56 of the Crozier Declaration is also found in the section "Configuring Security" the "JIRA 6 Documentation," as shown below.

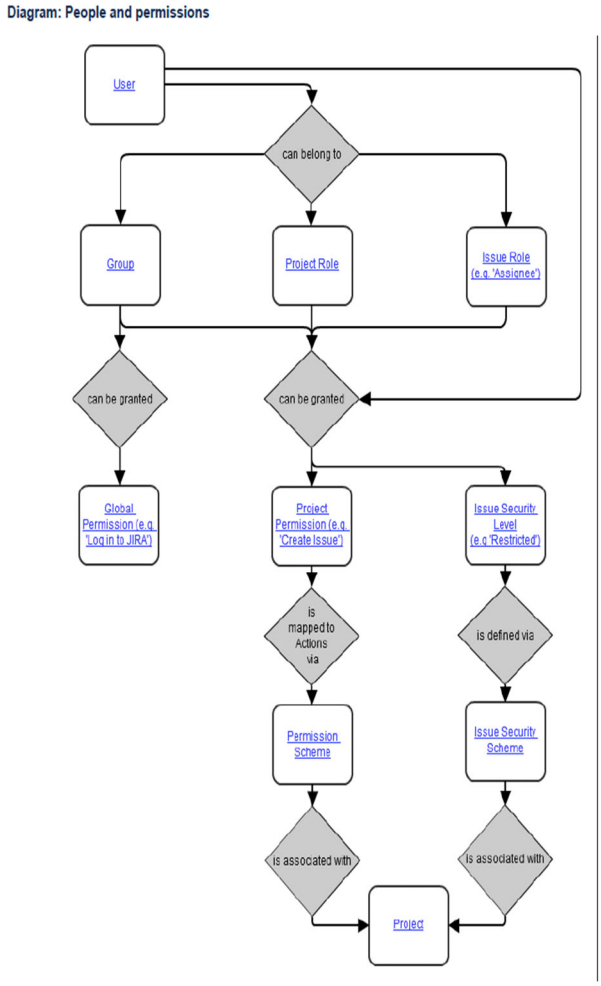
[redacted] (Moog)	JIRA 6 Documentation – Configuring Security (Publicly Available)
[redacted]	

Figure 16 – Use of Public Information Personnel in JIRA Guide

42. Mr. Crozier also fails to consider the similarities between Moog's documents and documents that were in Mr. Pilkington's possession before he joined Moog, and he provides no evidence that software Mr. Pilkington developed or information he had in his possession before he worked for Moog somehow belong to Moog. For example, the [redacted] (attached as Exhibit B8) found in the *MOOOG-04208.*A0016 container on the iDS environment at */Bravo 2 –* [redacted], which is before Mr. Pilkington joined Moog which I understand was in 2012. The earlier [redacted] document has many similarities to the Moog document [redacted], including, for example, a [redacted] on page 46 that is similar to diagram of [redacted] in Moog document [redacted]. Once again, not only is this information publicly known, but Mr. Pilkington was aware of, in possession of, and utilizing such information before he joined Moog. I understand this information also has been available to Mr. Pixley and Mr. Crozier given their access to the iDS environment but they do not address it in their declarations.

43. Analyzing Moog's [redacted] document shows that the information contained in that document, including specifically the examples selected by Mr. Crozier in his declaration describing the similarities between that document and the Skyryse document, JIRA Problem Reporting document (BIRD_SR_00001465), are either found exactly in or are closely related to information available in the public domain. I have reviewed the rest of the Moog's [redacted] and find that the use and reliance upon publicly available information continues throughout and do not find any information that is not attributable to the public domain

44. Moreover, beyond the publicly available information contained in Moog's [redacted], the information contained in the document is generally recognizable by someone experienced in the software development industry. In fact, I have seen such documentation and information many times. Furthermore, the information from the public domain that I have shown as examples above is not from obscure references, but from the documentation of Atlassian, the publisher of JIRA.

B. SVN Guide

45. Mr. Crozier's identification of an SVN guide (*Id*. at ¶¶25, 27) which he contends constitutes "Evidence of Misappropriation" by Skyryse. (*Id*. at ¶¶ 17, 22) is based on another example of Mr. Cozier claiming as Moog non-public information, information that is publicly available and generally known in the field.

46. For example, Mr. Crozier points to the Moog document, [redacted], which he asserts accompanies the Moog document [redacted],[17] in paragraphs 25, 27, and 45 of the Crozier Declaration. This document also includes publicly known information, including passages and concepts of Subversion ("SVN"), such as material from the popular SVN tool that I have used and seen used many times in the software development industry named TortoiseSVN manual,[18] or diagrams from Wikipedia,[19] as shown below.

---

[17] Found starting at page 108 of 2023.03.16 [400-6] [UNREDACTED] Exhibit A-10

[18] "Preface" TortoiseSVN. WEB. https://tortoisesvn.net/docs/release/TortoiseSVN_en/tsvn-preface.html. Accessed on April 8, 2023. (Ex. B9)

[19] "Apache Subversion" Wikimedia Foundation, Inc. WEB. https://en.wikipedia.org/wiki/Apache_Subversion. Accessed on April 18, 2023. (Ex. B10)

[redacted]

Figure 17 – [redacted]

What is TortoiseSVN?

TortoiseSVN is a free open-source Windows client for the *Apache™ Subversion®* version control system. That is, TortoiseSVN manages files and directories over time. Files are stored in a central *repository*. The repository is much like an ordinary file server, except that it remembers every change ever made to your files and directories. This allows you to recover older versions of your files and examine the history of how and when your data changed, and who changed it. This is why many people think of Subversion and version control systems in general as a sort of "time machine".

TortoiseSVN's Features

What makes TortoiseSVN such a good Subversion client? Here's a short list of features.

Shell integration

TortoiseSVN integrates seamlessly into the Windows shell (i.e. the explorer). This means you can keep working with the tools you're already familiar with. And you do not have to change into a different application each time you need the functions of version control.

And you are not limited to using the Windows Explorer; TortoiseSVN's context menus work in many other file managers, and also in the File/Open dialog which is common to most standard Windows applications. You should, however, bear in mind that TortoiseSVN is intentionally developed as an extension for the Windows Explorer. Thus it is possible that in other applications the integration is not as complete and e.g. the icon overlays may not be shown.

Icon overlays

The status of every versioned file and folder is indicated by small overlay icons. That way you can see right away what the status of your working copy is.

Figure 18 – Preface TortoiseSVN Guide

47. The organization of software development into different branches is a well-known concept in SVN. The Moog document, [redacted] documents this well-known concept with diagrams from publicly known information on Subversion ("SVN"), that can be found in Wikipedia,[20] as shown below.

[redacted]

Figure 19 – [redacted]

[20] "File:Revision controlled project visualization 28022019.svg" Wikimedia Foundation, Inc. WEB. https://commons.wikimedia.org/wiki/File:Revision_controlled_project_visualization_28022019.svg. Accessed on April 18, 2023. (Ex. B11)

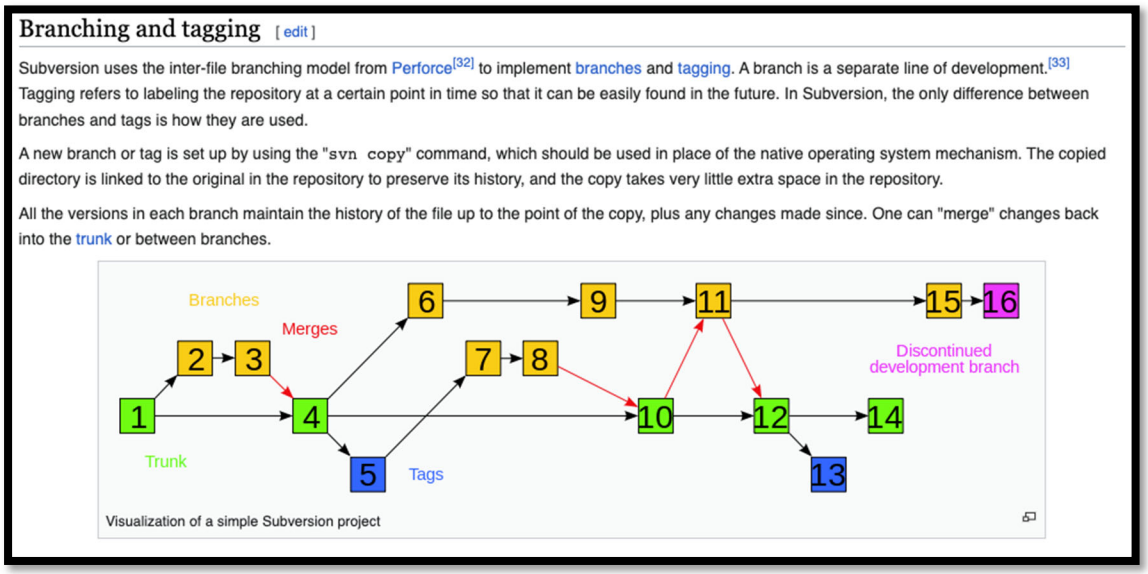


Figure 20 – Apache Subversion on Wikipedia

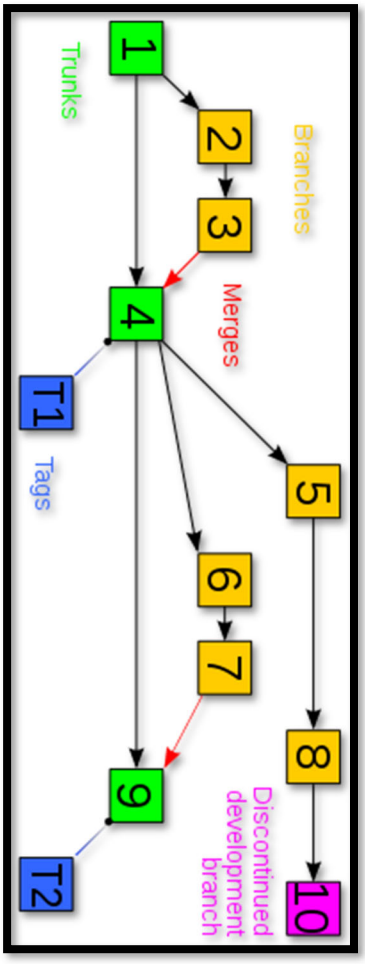


Figure 21 – Apache Subversion on Wikipedia

48. The interface of the third-party software TortoiseSVN is also publicly known. The Moog document, [redacted] documents this interface with images copied directly from TortoiseSVN's online manual,[21] as shown below.

[redacted]

Figure 22 – [redacted]

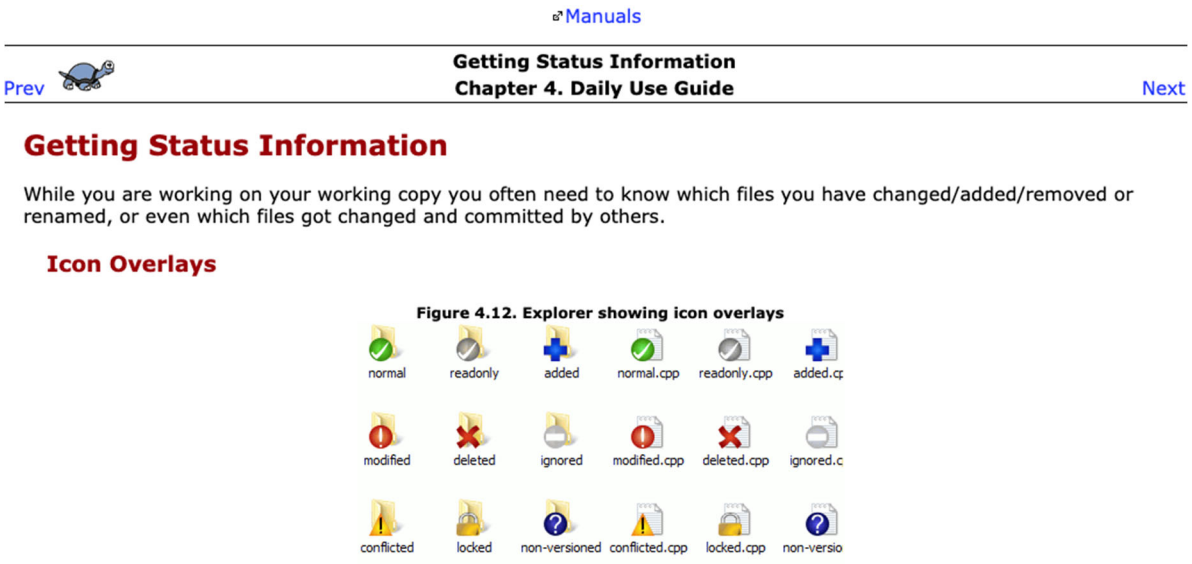


Figure 23 – TortoiseSVN Guide

[21] "Getting Status Information" TortoiseSVN. WEB. https://tortoisesvn.net/docs/release/TortoiseSVN_en/tsvn-dug-wcstatus.html. Accessed on April 8, 2023. (Ex. B12)

49. The organization of the development in different commits and branches is well known and the ability to visualize the versions in the third-party software TortoiseSVN is also publicly known. The Moog document, [redacted] documents this visualization with images copied directly from TortoiseSVN's online manual,[22] as shown below.

[redacted]

Figure 24 – [redacted]

[22] "Revision Graphs" TortoiseSVN. WEB. https://tortoisesvn.net/docs/release/TortoiseSVN_en/tsvn-dug-revgraph.html. Accessed on April 8, 2023. (Ex. B13)

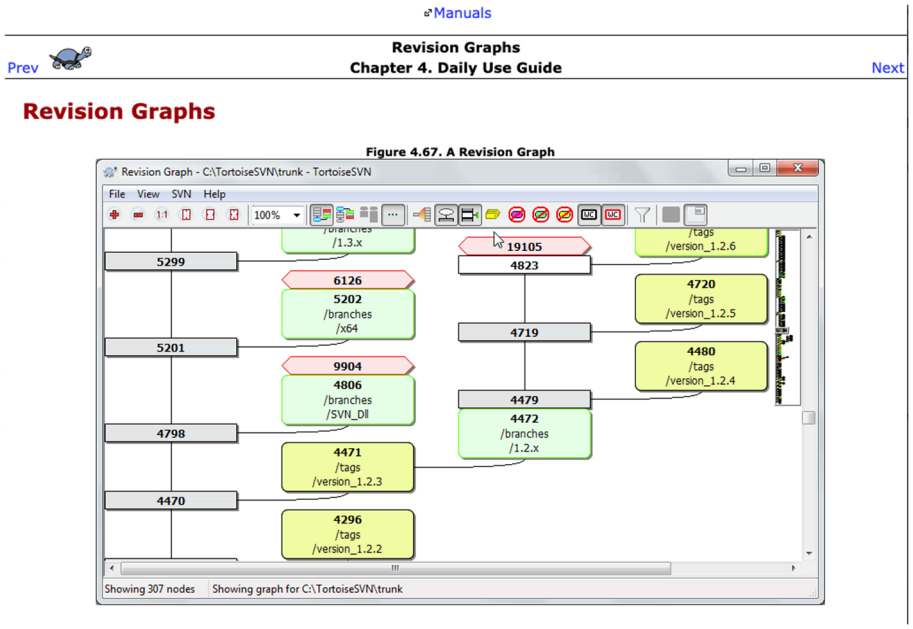


Figure 25 – TortoiseSVN Guide

50. I have reviewed the rest of the Moog document [redacted] and can confirm that it also uses and relies upon publicly available information. In fact, I could not find any information contained in that document that is not available in the public domain or would otherwise show that the information in this document is Moog's.

51. In addition to the above, I also found a copy of the book [redacted][24] This book was stored in the same directory as versions of the [redacted] document and includes much of the information that the Moog document is based on. This information, like the information described above, is also publicly known, and appears to have been within Mr. Pilkington's possession.

---

[23] "Version Control with Subversion For Subversion 1.7" Ben Collins-Sussman, Brian W. Fitzpatrick, C. Michael Pilato. WEB. https://svnbook.red-bean.com/en/1.7/svn-book.pdf Accessed on April 8, 2023.

[24] Found at [redacted]

52. Mr. Crozier also references at several points in his Declaration the use of a version control tool by Skyryse called Git, not Subversion or SVN. Git is a replacement for Subversion, and therefore these SVN documents would provide no utility to Skyryse, even if they included non-public information. Regardless, to the extent that Mr. Crozier or Moog make new assertions or provide new opinions about some use of this Moog document [redacted] or other SVN guides, I reserve the right to supplement or amend my opinion.

C. MDTE, SDTE, and ASTE Software

53. Mr. Crozier opines in his Declaration that "[b]ased on data provided for discovery at third party discovery vendor iDiscovery Solutions ('iDS'), Skyryse continues to use the Skyryse Desktop Environment (SDTE) test framework for its software testing activities." (Crozier Decl. ¶ 95.) He further claims that "[t]he SDTE framework is a nearly identical copy of the Moog Desktop environment (MDTE) test framework that is employed at Moog." (*Id.*) According to Mr. Crozier, his "analysis of the SDTE source code files show that they are nearly identical." (*Id.*) He further asserts that, based on "an on-site inspection of Skyryse's source code and Git repository," including "source code and Git repositories" made "available as of 4/15/20222," "evidence suggests that Skyryse is using the SDTE for software testing as of 4/15/2022." According to Mr. Crozier, this all constitutes "Evidence of Misappropriation and Use of Moog Data after March 11, 2022." (*Id.* at ¶ 49.)

54. As described further below, Mr. Crozier's opinions are flawed for several reasons, including because the evidence shows that the MDTE and SDTE source code are based on code developed by Mr. Pilkington prior to his employment with Moog, and I am not aware of any facts suggesting Moog owns such information. In addition, I have also reviewed Skyryse's current code base

and confirmed that the source code at the file paths Moog and its experts identified for SDTE code have been removed.

55. As stated above, Mr. Crozier's assertion that the SDTE and MDTE source code files are nearly identical fails to consider the origin of this source code. Specifically, there are two groups of files in MDTE. The first group are C and C++ files that are derived from, and are nearly identical to, a pre-existing collection of software found within a container on the iDS environment of materials that I understand came from Mr. Pilkington's personal computer. Specifically, the MDTE files are derived from, and largely identical to, the pre-existing Automated Software Test Environment ("ASTE"), with some reorganization of how the test specification information is organized and selected discussed below. I found many copies of these ASTE files throughout many containers in the iDS environment, which I understand was also available to Mr. Crozier and Mr. Pixley. I selected two as exemplary. This includes a collection of ASTE source code files and support documents in the container [redacted] on the iDS environment at the directory location [redacted] on the iDS environment in a zip compression file at the directory location [redacted]. This selection of documents, collectively, is attached as Exhibit B14.

56. Files in a zip compression file are particularly relevant, because a zip file preserves the original file metadata from the time that the zip file was created. Generally, as files are moved around computer file systems some of the file metadata, such as file creation and modification dates can get updated. However, a zip file essentially freezes this process, so that the file metadata, including timestamps is retained. As such, I can see that the version of ASTE files in the zip file originated in 2007, long before Mr. Pilkington joined Moog. In other words,

the existence of these ASTE files in a zip format establishes that these files originated from prior to Mr. Pilkington's employment at Moog.

57. In addition, the MDTE, SDTE, and ASTE source code files all have comments showing that Mr. Pilkington developed the original ASTE software as far back as 1997, as noted in the table mapping all three sets of files below. In fact, the ASTE files also include a header stating, "Copyright @ 2001, 2002 by Alin Pilkington," which predates his time at Moog, [redacted].

58. A careful comparison of the MDTE and SDTE files with the ASTE files strongly suggests the contents originated before Mr. Pilkington's tenure at Moog. The dated comments are arranged in the typical fashion that developers note development history, so there is no evidence to suggest these dates are not accurate. I understand that Mr. Pilkington was hired by Moog on July 30, 2012, but the dates in the very files on which Mr. Crozier relies show a record of development in 1996, 1997, and 2001, which Mr. Crozier fails to address or explain.

59. Mr. Crozier also discusses test data in Excel Spreadsheet SKY_IDS_0002190 and SKY_IDS_0002195 associated with this ASTE/MDTE/SDTE unit test software. (Crozier Decl. ¶¶117-118.) The Excel spreadsheet includes a sheet titled *RTB* with details of the unit tests and *Attributes* with further attributes of the unit test. However, Mr. Crozier fails to identify that the pre-existing ASTE software also included Excel Spreadsheets with nearly identical data. For example, the *RTB* and *Attributes* spreadsheets have the same structure as [redacted]

[redacted] and [redacted] and are found in the container [redacted] on the iDS environment. Again, these are just some examples of the many examples of ASTE source code and supporting files found across the [redacted] on the iDS environment that indicate that MDTE originated from a source other than Moog and is not necessarily Moog's information.

60. Mr. Crozier's declaration also refers to files called [redacted]. (Crozier Decl. at ¶ 52.) While the examples of ASTE source code I have reviewed to date do not include files with these filenames, the contents of these files are common, text manipulation and error reporting routines, that are derived from ASTE source code files [redacted] and [redacted] generally known concepts of converting datatypes,[25] and open source code. In fact, line 186 of [redacted] even says, "[redacted]." As such, the contents of these files cannot be distinguished from the public domain or pre-existing ASTE source code, and do not appear to have originated at Moog.

61. Mr. Crozier also refers to certain Python files related to MDTE. Python is a software programming language that is often thought of as more user friendly language that is often used for creating scripts to quickly automate everyday tasks. I have reviewed these files and they use publicly available Python libraries[26] and techniques[27] to implement a Graphical User Interface ("GUI") to run the core MDTE program made from the C and C++ files, all of which came from

---

[25] "libc.c" University of Minnesota. WEB. https://www-users.cse.umn.edu/~smccaman/assign/rewrite-2013/libc.c. Accessed on April 10, 2023.

[26] "PySimpleGui" Python Software Foundation. WEB. https://pypi.org/project/PySimpleGUI/. Accessed on April 10, 2023.

[27] "How do i print results to the command line instead of a popup window using PySimpleGUI?" Stack Exchange Inc. https://stackoverflow.com/questions/55910191/how-do-i-print-results-to-the-command-line-instead-of-a-popup-window-using-pysim. Accessed on April 8, 2023.

the ASTE software. As stated, I have not been able to review every version of ASTE on the iDS environment given the time constraints, but regardless of when this GUI originated, the GUI is nothing more than a graphical interface that does not change the core unit testing operations of the C and C++ code that already existed in the ASTE software. Furthermore, the GUI is built with generally known concepts and libraries, such as gathering input parameters[28] and passing the parameters to an executable.[29]

62. In sum, Mr. Crozier's opinions suffer from multiple flaws, the most striking being that he does not appear to have considered the history of the MDTE files, which were clearly derived from and nearly identical to the ASTE software that predates Mr. Pilkington's employment with Moog. As such, Mr. Crozier has failed to properly distinguish the MDTE source code from the ASTE software and has therefore failed to properly identify what, if any, portion of the MDTE source code is unique or proprietary to, or owned by, Moog.

Moog MDTE C & C++ Files	**Skyryse SDTE C & C++ Files**	**Previously Existing Automated Software Test Environment ("ASTE") (combined as Exhibit B14)**
[redacted]	CT_CreateTestSpec.c SKY_IDS_0001654 Includes comments showing earlier origin: • 02/06/97 A. Pilkington Initial coding" • 11/21/01 A. Pilkington (1)	[redacted]

[28] "How do i print results to the command line instead of a popup window using PySimpleGUI?" Stack Exchange Inc. WEB. https://stackoverflow.com/questions/55910191/how-do-i-print-results-to-the-command-line-instead-of-a-popup-window-using-pysim. Accessed on April 8, 2023.

[29] "Devenv command-line switches" Microsoft. WEB. https://learn.microsoft.com/en-us/visualstudio/ide/reference/devenv-command-line-switches?view=vs-2022 . Accessed on April 8, 2023.

[redacted]	Updated with new coding standards	[redacted]
	CT_CreateTestSpec.h SKY_IDS_0001665 Includes comments showing earlier origin: • 09/26/96 A. Pilkington Initial coding"	

[redacted]		[redacted]
	SS_Spreadsheet.c SKY_IDS_0001668 Includes comments showing earlier origin: • 09/26/96 A. Pilkington Initial coding	
	SS_Spreadsheet.h SKY_IDS_0001727 Includes comments showing earlier origin: • 09/26/96 A. Pilkington Initial coding	

[redacted]

TS_TestServices.c
SKY_IDS_0001731

Includes comments showing earlier origin:

- 11/09/01 A. Pilkington Initial coding for C/C++ generation

TS_TestServices.h
SKY_IDS_0001909

Includes comments showing earlier origin:

- 09/26/96 A. Pilkington Initial coding

[redacted]

[redacted]

Common.c
SKY_IDS_0001912

[redacted]

[redacted]	Common.h SKY_IDS_0001949	[redacted]
	IO_TestInOut.cpp SKY_IDS_0001956 Includes comments showing earlier origin: • 11/09/01 A. Pilkington Initial coding for C/C++ generation	

[redacted]	IO_TestInOut.h SKY_IDS_0002002 Includes comments showing earlier origin: • 02/06/97 A. Pilkington Initial coding" • 11/21/01 A. Pilkington (1) Updated with new coding standards	[redacted]
	ReadMe.txt SKY_IDS_0002005	
	SDTE.h SKY_IDS_0002009 Includes comments showing earlier origin: • 04/06/02 A. Pilkington Initial coding	

[redacted]		[redacted]
[redacted]	SD_SystemDef.h SKY_IDS_0002006 Includes comments showing earlier origin: • 01/13/97 A. Pilkington Initial coding • 11/21/01 A. Pilkington (1) Updated with new coding standards	[redacted]
[redacted]	SI_SysInfo.h SKY_IDS_0002013 Includes comments showing earlier origin: • 01/13/97 A. Pilkington Initial coding	[redacted]

[redacted]	• 11/26/01 A. Pilkington (1) Updated with new coding standards	[redacted]
	TD_TestDriver.cpp SKY_IDS_0002016 Includes comments showing earlier origin: • 11/09/01 A. Pilkington Initial coding for C/C++ generation	
	TT_TestTypes.h SKY_IDS_0002069 Includes comments showing earlier origin: • A. Pilkington 12 Sep 96	

[redacted]

Added the ability to collect timing info

- A. Pilkington 12 Sep 96 Added double precision floating point
- A. Pilkington 18 Oct 96 Added multi-rate capability
- M. Kim 23 Oct 96 (1) added capability for threshold in %
- A. Pilkington 14 Dec 96 (1) Added capability to selectively 'jam' in new values
- A. Pilkington 06 Jan 97 Corrrected Variant record types so that initialized
- A. Pilkington 12/26/01 (1) Updated with new coding standards

[redacted]

1. RBT Spreadsheets

63. At paragraphs 10-13 of the Pixley Declaration, Mr. Pixley discusses certain "RBT Spreadsheets" he claims are associated with MDTE. Mr. Pixley does not specify where he found the RBT Spreadsheets, but the screenshot shows an example at [redacted], which despite Mr. Pixley's lack of specificity appears to be one of the RBT Spreadsheets to which he refers. Mr. Pixley relies on these files to assert that Skyryse has been using the SDTE code after March 11, 2022. But Mr. Pixley again ignores the evidence suggesting that MDTE originated from a source other than Moog and is not necessarily Moog's information.

64. Beyond that, and more significantly, the contents of [redacted] file related to MDTE follow the same format and design found in pre-existing ASTE support files that I discussed above. They include, for example, the same rows [redacted] Therefore, Mr. Pixley, like Mr. Crozier, is pointing to files that originated with the pre-existing ASTE software that no evidence I'm aware of suggests is actually Moog's. In addition, Mr. Pixley states that the RBT Spreadsheets appeared be derived from a shared template. I agree, except that Mr. Pixley fails to recognize that the original template goes back to the ASTE software that existed before Mr. Pilkington joined Moog.

65. Since Mr. Pixley did not identify the RBT Spreadsheet files he examined, or provide their location I cannot verify his findings regarding authorship or file header metadata. However, I did search the [redacted] directory and found 64 spreadsheet files, of which 51 listed Robert A Pilkington as the author and none listing Eric Chung, which appear to be a discrepancy in Mr. Pixley's findings. To the extent that Mr.

Pixley specifies where he found the RBT files that he allegedly reviewed, I reserve the right to supplement or amend my opinion.

D. RTOS Files

66. As described above, Mr. Crozier also opines that his "analysis of SRTOS html files which are eventually compiled to the .chm file (chm file is a compressed html file) shows that it contains numerous identical or slightly modified figures (ie.: SRTOS replaces eRTOS), identical document structure and number word-for-word passages to Moog eRTOS.chm files." (*Id*. at ¶ 103.) Mr. Crozer opines that "[t]his preliminary design document along with source code provided during discovery suggest that the Skyryse SRTOS operating system is copied directly from the Moog eRTOS operating system." (*Id*.) Mr. Crozier claims that these findings also constitute "Evidence of Misappropriation and Use of Moog Data after March 11, 2022." (*Id*. at ¶ 49.)

67. Although he vaguely refers to "source code provided during discovery," Mr. Crozier does not identify any Moog eRTOS code that would form the basis of this opinion, nor does he identify any analysis of that code, or compare any eRTOS code with any SRTOS code. Instead, Mr. Crozier attempts to rely on design documents to form the basis of his opinion that Moog and Skyryse's real time operating systems are identical. But this basis is flawed because that conclusion can only be reached by reviewing and comparing the source code for the two programs, which analysis [redacted] (Crozier Dep. 113:23-114:1; 116:12-14.) Neither Moog nor Mr. Crozier have identified the source code for eRTOS. However, I have been able to find source code on devices provided to iDS that appear to contain eRTOS code, such as in the [redacted] container in the iDS environment. That code does not appear to have been the source of the SRTOS code Mr. Crozier opines on, for a number of reasons, including that the eRTOS code files are different from the SRTOS code files. This is not surprising,

because a real time operating system by its very nature has to be highly customized and specific to the product to which it applies. In addition, I have also reviewed Skyryse's current code base and confirmed that the source code at the file paths Moog and its experts identified for SRTOS code has been removed.

68. Finally, I have identified earlier versions of RTOS code on Mr. Pilkington's personal computer that predates his time at Moog, and I have seen no evidence that this code is Moog's. For example, the source code file [redacted], in the [redacted] container in the iDS environment, which includes a number of indicators that the code pre-dates Mr. Pilkington's time at Moog and may be third-party code of Lear Astronics, includes lines of source code that are identical to the eRTOS file [redacted] also in the [redacted] container in the iDS environment. While neither Moog nor Mr. Crozier identified any eRTOS source code, the file [redacted] file is referenced in the Moog eRTOS design document Mr. Crozier relies upon. Neither Moog nor Mr. Crozier mentions, let alone addresses, these prior third-party versions of RTOS. While I discovered this pre-existing code too late to review the full extent of similarities to eRTOS, its existence undermines Mr. Crozier's (and Moog's) claims that eRTOS constitutes Moog non-public information. I found other non-Moog documents, including documents from Mr. Pilkington that pre-date his time at Moog, on the iDS devices that disclose the various concepts related to RTOS that Mr. Crozier relies on.

69. As discussed above, Real-Time Operating Systems ("RTOS") are also widely used and generally known in the public domain.[30] An RTOS is a type of operating system that is generally implemented to be small, efficient, and tightly

[30] "What Is a Real-Time Operating System (RTOS)?" Wind River Systems, Inc., WEB. https://www.windriver.com/solutions/learning/rtos. Accessed on April 12, 2023.

bound to timed operations. They are often used in embedded systems to provide the scheduling, performance, reliability, and exact timing support for the specific tasks of a system instead of the more general, but noncritical, operations of a conventional computer. RTOS is a commonly known concepts in software development and real-time systems and would be expected in systems that require reliability and time critical operations, such as avionics systems. Given this, Mr. Crozier's assertions that there was continued development of an RTOS system are vague as to what elements of import, if any, could be Moog's and not available from other sources.

70. Furthermore, the directories to which Mr. Crozier points include third-party code, including for example at [redacted] container from the IDS environment with copyright headers for the third-party Texas Instruments, Inc. Mr. Crozier does not specify what files or actual content of sRTOS he believes is correlated to eRTOS. However, these directories that Mr. Crozier points to include board support package ("BSP") files. To the extent that Mr. Crozier is alleging BSP files are purportedly Moog non-public information, their reliance on publicly available information is discussed below in conjunction with my response to Mr. Crozier's discussion of BSP files.

71. I note as well, that the Skyryse-SRTOS files are found with metadata of a Git repository and not a SVN repository at, for example [redacted] container from the IDS environment.

E. Skyryse Doxygen Python Scripts

72. Section III.F at paragraphs 100 – 103 of the Crozier Declaration discusses Moog Python scripts[31] and Skyryse Python Scripts,[32] which Mr. Crozier asserts are found in a Google drive for generating source code documentation with the third party tool Doxygen and relate to HTML help files.[33] Mr. Crozier's assertions are again vague in what elements of import, if any, he contends are purportedly Moog non-public information.

73. As discussed above, Python is a software programming language that is often thought of as more user friendly programming language that is often used for creating scripts to quickly automate everyday tasks rather than creating full applications. Developers often create scripts, such as those written in Python, to automate repetitive tasks involving searching, collecting, and formatting data that becomes tedious to preform manually. Asl also discussed above, Doxygen is a well-known open source program for generating HTML documentation from source code.

74. Mr. Crozier lists 10 Python files at paragraph 101 of his Declaration, which he claims "are property of Moog." But Mr. Crozier fails to identify that these files include publicly available open-source code that is not Moog's. For example, the source code file *SKY_00000250* is an open source file named [redacted] The file [redacted] can be found online as part of the open source Python Script for Notepad++ project.[34] Notepad++ is a popular text editor.[35] By including such obvious open source files that are publicly available and not

---

[31] Found at 2023.03.16 [400-6] [UNREDACTED] Exhibit F-3
[32] Found at 2023.03.16 [400-6] [UNREDACTED] Exhibit F-2
[33] [redacted]
[34] "Python Script for Notepad++" GitHub, Inc. WEB. https://github.com/bruderstein/PythonScript/blob/bfef71f9772a528905dd03931a949f460c977159/scripts/startup.py. Accessed on April 11, 2023.
[35] "What is Notepad++" Don Ho. WEB. https://notepad-plus-plus.org/. Accessed on April 11, 2023.

Moog's, Mr. Crozier has failed to distinguish what element of the Doxygen Python scripts could qualify as Moog non-public information.

75. Along with the open source file, six of the Doxygen Python scripts identified by Mr. Crozier essentially repeats a single method [redacted] and are therefore void of substantial information. These include SKY_00000122, SKY_00000128, SKY_00000134, SKY_00000140, SKY_00000145, and SKY_00000211.

76. The three remaining files SKY_00000148, SKY_00000179, and SKY_00000219 include nearly identical implementations of [redacted], which is described in the comments at the beginning of each source code file as "[redacted]." The following discussion of the public availability of these files therefore relates to all three versions.

77. As discussed above, the Doxygen tool is open source software described as "the de facto standard tool for generating documentation from annotated C++ source"[36] As such, Doxygen and its use for generating documentation from annotated source code are well known concepts in the public domain.

78. The term SDD format is an industry standard term for Software Design Description ("SDD"). SDD is a formal mechanism for describing, and thereby dictating, the design of Computer Software Configuration Item (CSCI), which is one or more pieces of software that supports a particular functionality on a single computer system. The SDD format and its usage is defined in detail in publicly available government publications such as DI-IPSC-81435A.[37] For example, the DID DI-IPSC-81435A specifies the topics, descriptions, formatting,

---

[36] "Doxygen" Dimitri van Heesch. WEB https://www.doxygen.nl/index.html. Accessed on April 11, 2023.
[37] http://www.tc.faa.gov/its/worldpac/Standards/dids/DI-IPSC-81435A.doc

and organization of documentation of Computer Software Configuration Items ("CSCI"). The SDD, therefore, dictates how software should be documented, which is what is being followed in the Doxygen Python scripts.

DATA ITEM DESCRIPTION

Title: Software Design Description (SDD)

Number: DI-IPSC-81435A
AMSC Number: N7360
DTIC Applicable: No
Office of Primary Responsibility:
Applicable Forms: N/A

Approval Date: 15 December 1999
Limitation: N/A
GIDEP Applicable: No

Use/relationship:

The Software Design Description (SDD) describes the design of a Computer Software Configuration Item (CSCI). It describes the CSCI-wide design decisions, the CSCI architectural design, and the detailed design needed to implement the software. The SDD may be supplemented by Interface Design Descriptions (IDDs) (DI-IPSC-81436) and Database Design Descriptions (DBDDs) (DI-IPSC-81437) as described below.

The SDD, with its associated IDDs and DBDDs, is used as the basis for implementing the software. It provides the acquirer visibility into the design and provides information needed for software support.

Figure 26 – Publicly Available Software Design Description from DI-IPSC-81435A

79. Mr. Crozier failed to identify that these Doxygen Python scripts that he claims "are property of Moog" include publicly available open source software to generate documentation according to publicly available specifications. Mr. Crozier and Moog have therefore failed to distinguish what, if any, elements of these Doxygen Python scripts they claim are owned by Moog are not based on publicly available information.

F. Skyryse sRTOS Design Document

80. Section III.F, Paragraphs 104-109 of the Crozier Declaration also discusses RTOS design documentation, including what Mr. Crozier describes as

Moog's eRTOS Design Document,[38] and a Skyryse sRTOS Design Document.[39] Mr. Crozier's assertions are again vague in what elements of import, if any, are not publicly available information. And Mr. Crozier does not provide any analysis of the substance of either document, let alone a comparison of that substance. Instead, his opinion is limited to comparing the document structure and text. But he does not address that the documents are general discussions of Real-Time Operating Systems ("RTOS") including the Board Support Package ("BSP"), which supports the interface with actual hardware and an Application Programming Interface ("API"), which supports the interface with applications in a system. These elements are described as a Software Configuration Item ("CSCI") in the Software Design Document format, which is dictated by publicly available specifications, so I would expect similarities as both documents adhere to these third party, publicly available specifications.

81. As discussed above, Real-Time Operating Systems ("RTOS") are also widely known in the public domain.

82. Likewise, APIs and BSPs are widely known concepts and categories of software.[40]

83. The design elements, such as [redacted] discussed in these Moog and Skyryse RTOS design documents are not only general to RTOS technology, but are specifically discussed for aviation system design publications, such as the Technical Standard for Future Airborne Capability Environment FACE, Edition 2.1.1,[41] shown below. This document, for example, outlines

---

[38] Found at 2023.03.16 [400-6] [UNREDACTED] Exhibit F-5
[39] Found at 2023.03.16 [400-6] [UNREDACTED] Exhibit F-4
[40] "Lecture 2 Platforms RTOS" University of Texas at El Paso. WEB. https://www.cs.utep.edu/isalamah/courses/5372/CS5372-Lecture-2.pdf. Accessed on April 24, 2023. (Ex. B15)
[41] "Technical Standard for Future Airborne Capability Environment FACE, Edition 2.1.1" The Open Group FACETM Consortium. WEB. https://www.opengroup.org/face/tech-standard-2.1. Accessed on April 11, 2023. (Ex. B16)

software resources, such as API and BSP that are expected in a CSCI. Mr. Crozier does not make any attempt to distinguish the text he points to from these same general design concepts.

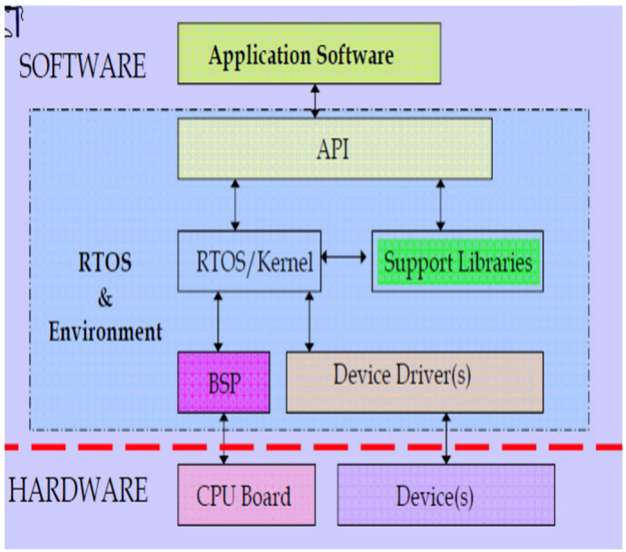
Moog's [redacted]	Publicly Available Lecture 2 Platforms RTOS
[redacted]	

Figure 27 – Use of Public Information Personnel in [redacted]

The APIs of programming language run-times and application frameworks are managed in a similar way to the FACE Operating System Profile API sets. To be specific, only the APIs provided by operating systems, programming language run-times, and application frameworks that are available for use by FACE components are managed. These APIs are often referred to as the "top-side interfaces" due to their typical placement in layered architectural diagrams. See Section 3.12 for FACE defined programming language run-times, and Section 3.13 for application frameworks approved for use in FACE conformant solutions.

Programming language run-times and application frameworks that do not provide a FACE approved API set (i.e., top-side interfaces) may be used in FACE conformant solutions only when included as part of a FACE UoP along with its client components. Programming language run-times and/or application frameworks that do not provide a FACE conformant API set are never part of the OSS.

The firmware and software resources (*a.k.a.* "bottom-side interfaces") used by operating systems, programming language run-times, and application frameworks are expected to vary and are not prescribed or otherwise governed by the FACE Technical Standard. As an example, operating systems are often fielded on computing hardware with different Board Support Packages (BSPs) and Chip Support Packages (CSPs). Operating systems use device drivers to interface with varying dissimilar hardware devices. Combining these examples, the pattern of providing a standardized API set to software components while simultaneously supporting the wide variance of BSPs, CSPs, and Drivers may be considered commonplace. Similarly, the bottom-side interfaces of programming language run-times and application frameworks may vary. The program-specific choice of an operating system may constrain the choice of a programming language run-time implementation. The choice of an application framework such as OSGi that executes on a programming language run-time is constrained by the choice of programming language run-time.

The bottom-side interfaces of FACE defined programming language run-times and application frameworks are not constrained to use a specific FACE operating systems API profile. In other words, if the choice for a FACE conformant operating systems implementation is a full commercial version that provides APIs beyond the approved FACE profile, the programming language run-times and application frameworks defined by FACE may use the additional non-conformant FACE APIs. In much the same way that this practice preserves the ability to use device drivers and non-conformant FACE applications that have been fielded on the selected commercial version of an operating system, commercial versions of fielded programming language run-times and application frameworks do not require modification. This practice does not add additional cost, effort, latency, or "time to market" to the existing commercial products as well as preserving the size of the existing marketplace which fosters competition and innovation.

Figure 28 – Page 52 of Technical Standard for Future Airborne Capability Environment FACE, Edition 2.1.1

84. Furthermore, the organization of these documents follows industry standard data item description ("DID") for Software Design Description ("SDD"),

such as DI-IPSC-81435A,[42] which specify the topics, descriptions, formatting, and organization of documentation of Computer Software Configuration Items ("CSCI"). As shown below, the *DI-IPSC-81435A* specification, for example, instructs the inclusion of sections of the [redacted] that Mr. Crozier identifies in paragraphs 104-109 of the Crozier Declaration. It is not surprising that these publicly known and specified elements are found in the Moog document, since page 17 of the Moog's [redacted]."

[42] "DI-IPSC-81435A." WEB. http://www.tc.faa.gov/its/worldpac/Standards/dids/DI-IPSC-81435A.doc. Accessed on April 11, 2023. (Ex. B17)

1.1 Identification. This paragraph shall contain a full identification of the system and the software to which this document applies, including, as applicable, identification number(s), title(s), abbreviation(s), version number(s), and release number(s).

1.2 System overview. This paragraph shall briefly state the purpose of the system and the software to which this document applies. It shall describe the general nature of the system and software; summarize the history of system development, operation, and maintenance; identify the project sponsor, acquirer, user, developer, and support agencies; identify current and planned operating sites; and list other relevant documents.

1.3 Document overview. This paragraph shall summarize the purpose and contents of this document and shall describe any security or privacy considerations associated with its use.

2. Referenced documents. This section shall list the number, title, revision, and date of all documents referenced in this document. This section shall also identify the source for all documents not available through normal Government stocking activities.

3. CSCI-wide design decisions. This section shall be divided into paragraphs as needed to present CSCI-wide design decisions, that is, decisions about the CSCI's behavioral design (how it will behave, from a user's point of view, in meeting its requirements, ignoring internal implementation) and other decisions affecting the selection and design of the software units that make up the CSCI. If all such decisions are explicit in the CSCI requirements or are deferred to the design of the CSCI's software units, this section shall so state. Design decisions that respond to requirements designated critical, such as those for safety, security, or privacy, shall be placed in separate paragraphs. If a design decision depends upon system states or modes, this dependency shall be indicated. Design conventions needed to understand the design shall be presented or referenced. Examples of CSCI-wide design decisions are the following:

a. Design decisions regarding inputs the CSCI will accept and outputs it will produce, including interfaces with other systems, HWCIs, CSCIs, and users (4.3.x of this DID identifies topics to be considered in this description). If part or all of this information is given in Interface Design Descriptions (IDDs), they may be referenced.

Figure 29 – DID DI-IPSC-81435A

4.3 Interface design. This paragraph shall be divided into the following subparagraphs to describe the interface characteristics of the software units. It shall include both interfaces among the software units and their interfaces with external entities such a systems, configuration items, and users. If part or all of this information is contained in Interface Design Descriptions (IDDs), in section 5 of the SDD, or elsewhere, these sources may be referenced.

4.3.1 Interface identification and diagrams. This paragraph shall state the project unique identifier assigned to each interface and shall identify the interfacing entities (software units, systems, configuration items, users, etc.) by name, number, version, and documentation references, as applicable. The identification shall state which entities have fixed interface characteristics (and therefore impose interface requirements on interfacing entities) and which are being developed or modified (thus having interface requirements imposed on them). One or more interface diagrams shall be provided, as appropriate, to depict the interfaces.

4.3.x (Project-unique identifier of interface). This paragraph (beginning with 4.3.2) shall identify an interface by project-unique identifier, shall briefly identify the interfacing entities, and shall be divided into subparagraphs as needed to describe the interface characteristics of one or both of the interfacing entities. If a given interfacing entity is not covered by this SDD (for example, an external system) but its interface characteristics need to be mentioned to describe interfacing entities that are, these characteristics shall be stated as assumptions or as "When [the entity not covered] does this, [the entity that is covered] will ...." This paragraph may reference other documents (such as data dictionaries, standards for protocols, and standards for user interfaces) in place of stating the information here. The design description shall include the following, as applicable, presented in any order suited to the information to be provided, and shall note any differences in these characteristics from the point of view of the interfacing entities (such as different expectations about the size, frequency, or other characteristics of data elements):

Figure 30 – DID DI-IPSC-81435A

Figure 31- Page 17 of [redacted], [UNREDACTED] Exhibit F-5

85. The Moog [redacted]. The DO-178B/C is the *Software Considerations in Airborne Systems and Equipment Certification*, from RTCA, Inc.,[43] [44] which provides guidance for determining whether software complies with airborne systems requirements. [redacted] and includes information from page 19 of the DO-178B document, excerpted as Exhibit B18. I found multiple copies of this the [redacted] document amongst Mr. Pilkington's files, named [redacted], which indicates Mr. Pilkington had this document in October 2011, before he joined Moog.[45]

---

[43] Software Considerations In Airborne Systems and Equipment Certification," RTCA, Inc. WEB. https://www.rtca.org/training/do-178c-training/. Accessed April 12, 2023.
[44] Software Considerations In Airborne Systems and Equipment Certification," DOO-178B.
[45] Found in the [redacted]

[redacted]

Figure 32 – [redacted]

SOFTWARE DEVELOPMENT PROCESS

This section discusses the objectives and activities of the software development processes. The software development processes are applied as defined by the software planning process (section 4) and the Software Development Plan (subsection 11.2). Table A-2 of Annex A is a summary of the objectives and outputs of the software development processes by software level. The software development processes are:

- Software requirements process.
- Software design process.
- Software design process.
- Integration process.

Software development processes produce one or more levels of software requirements. High-level requirements are produced directly through analysis of system requirements and system architecture. Usually, these high-level requirements are further developed during the software design process, thus producing one or more successive, lower levels of requirements. However, if Source Code is generated directly from high-level requirements, then the high-level requirements are also considered low-level requirements. and the guidelines for low-level requirements also apply.

The development of a software architecture involves decisions made about the structure of the software. During the software design process, the software architecture is defined and low-level requirements are developed. Low-level requirements are software requirements from which Source Code can be directly implemented without further information.

Figure 33 – DO-178

86. The examples that Mr. Crozier provides of Moog's eRTOS Design Document[46] and Skyryse sRTOS Design Document contain information that is publicly available and generally known in software design and development. The documentation also follows the expect SDD format expected in the aviation

[46] Found at 2023.03.16 [400-6] [UNREDACTED] Exhibit F-5

industry. But Mr. Crozier does not actually provide any evidence or analysis to link the design documents to any Moog or Skyryse source code files. Mr. Crozier has also failed to perform any analysis to distinguish this RTOS information from what is generally known and dictated by the industry.

G. Skyryse BSP Files

87. Section III.G of the Crozier Declaration, at paragraphs 110-116, discusses board support files ("BSP"), HB0000094, 87, 77, 73, 56, 51, 28, 22, 9.[47] To the extent that RTOS and BSP are generally known concepts in software development and real-time systems, Mr. Crozier's assertions are vague in what elements of import, if any, he thinks qualify as Moog non-public information.

88. As an initial matter, neither Mr. Crozier nor Moog assert that Skyryse is using the same hardware as Moog. This matters because, as Mr. Croizer admits, "BSP is the low level driver layer of an operation system." (Crozier Decl. ¶ 110.) In other words, the BSP source code is hardware-specific, and different hardware requires different BSP code. Thus, any purported evidence that Skyryse is working on BSP code would not establish that Skyryse is using any of Moog's information in developing that code.

89. Furthermore, it is unclear what relevance Mr. Crozier places on an alleged statement by Skyryse personnel David Lee that "[f]or now, we are focusing on AM5728 (FCC1)." AM5728 is a processor,[48] which is a publicly known and published technology from the third-party supplier Texas Instruments. Therefore, knowledge or use of a Texas Instruments AM5728 is a processor would not be Moog non-public information.

---

[47] Found at 2023.03.16 [400-6] [UNREDACTED] Exhibit G-2

[48] "AM5728" Texas Instruments Incorporated. WEB. https://www.ti.com/product/AM5728#product-details. Accessed April 12, 2023.

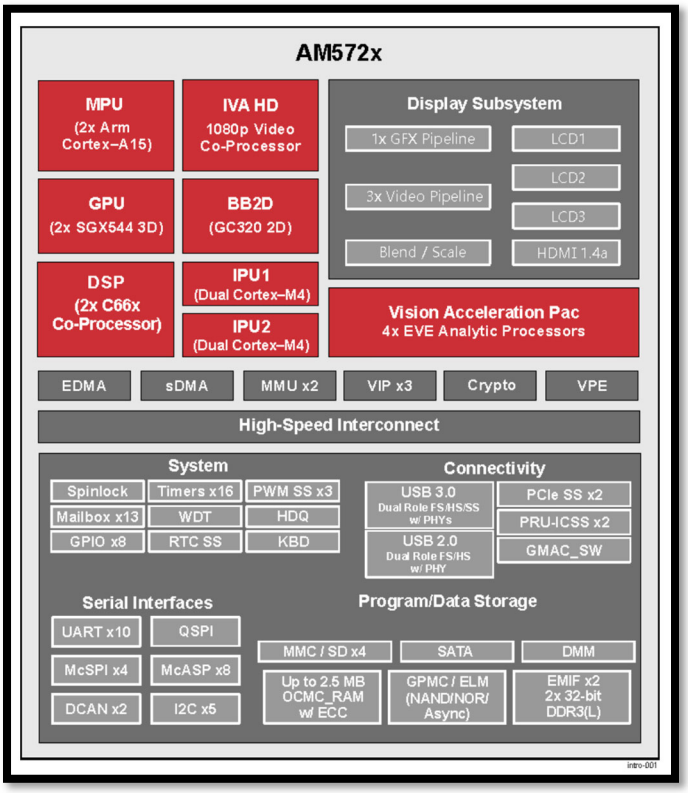


Figure 34 – Texas Instruments AM572x Block Diagram

90. Furthermore, one of the BSP files that Mr. Crozier identified states that it includes functions for the "A53 processor on the i.MX 8M Mini System On Module (SOM)." The I.MX8M Mini System on Module is a hardware board from SolidRun that includes a NXP's Arm Cortex A53 processor.[49] Since this identified BSP software clearly states that it associates with this MX8M hardware and Skyryse developers are discussing the Texas Instruments AM5728 processor, it is not surprising that sections of the identified BSP source code can be found online. For example, the function ***BSP_CPU_ClockSetRootMux*** includes the same

[49] "I.MX8M Mini System on Module" SolidRun, Inc. WEB. https://www.solid-run.com/embedded-industrial-iot/nxp-i-mx8-family/imx8m-mini-som/#overview. Accessed on April 12, 2023.

instructions as a source code file from NXP that can be found on GitHub as ccm_imx7d.h,[50] [51] [52] and pca9535.c.[53]

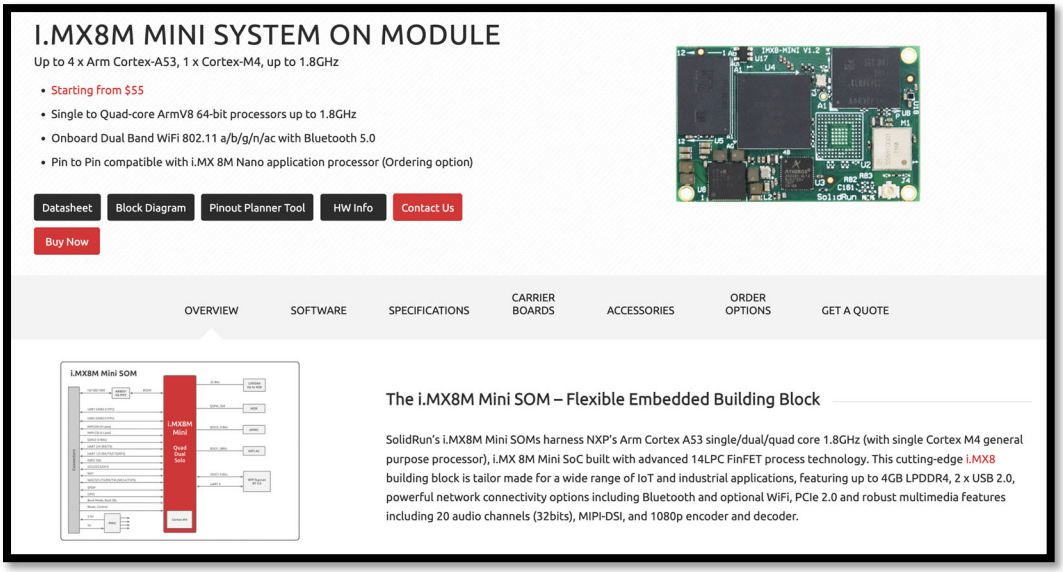


Figure 35 – I.MX8M Mini System on Module

[50] "ccm_imx7d.h" GitHub, Inc. https://github.com/VirgilYP/peng/blob/94451b22fcc6a068de2a2982b202772c6bbefae8/ext/hal/nxp/imx/drivers/ccm_imx7d.h Accessed on April 12, 2023. (Ex. B19)

[51] "ccm_imx7d.c" GitHub, Inc. ttps://github.com/VirgilYP/peng/blob/94451b22fcc6a068de2a2982b202772c6bbefae8/ext/hal/nxp/imx/drivers/ccm_imx7d.c Accessed on April 12, 2023. (Ex. B20)

[52] "MIMXRT1052.h" Github, Inc. https://raw.githubusercontent.com/ARMmbed/mbed-os/master/targets/TARGET_NXP/TARGET_MCUXpresso_MCUS/TARGET_MIMXRT1050/device/MIMXRT1052.h. Accessed on April 12, 2023.

[53] "pca9535.c" Github, Inc. https://github.com/yulei/amk/blob/b6d1333a174574083a43989e9805be4300862286/main/drivers/pca9535.c. Accessed on April 12, 2023. (Ex. B21)

Figure 36 – 2023.03.16 [400-6] [UNREDACTED] Exhibit G-2

```
355  static inline void CCM_SetRootMux(CCM_Type * base, uint32_t ccmRoot, uint32_t mux)
356  {
357      CCM_REG(ccmRoot) = (CCM_REG(ccmRoot) & (~CCM_TARGET_ROOT_MUX_MASK)) |
358                                  CCM_TARGET_ROOT_MUX(mux);
359  }
```

Figure 38 – NXP ccm_imx7d.h

Figure 39 – 2023.03.16 [400-6] [UNREDACTED] Exhibit G-2

```
43  void CCM_SetRootDivider(CCM_Type * base, uint32_t ccmRoot, uint32_t pre, uint32_t post)
44  {
45      assert (pre < 8);
46      assert (post < 64);
47
48      CCM_REG(ccmRoot) = (CCM_REG(ccmRoot) &
49                          (~(CCM_TARGET_ROOT_PRE_PODF_MASK | CCM_TARGET_ROOT_POST_PODF_MASK))) |
50                         CCM_TARGET_ROOT_PRE_PODF(pre) | CCM_TARGET_ROOT_POST_PODF(post);
51  }
```

Figure 40 – NXP ccm_imx7d.c

Figure 41 – 2023.03.16 [400-6] [UNREDACTED] Exhibit G-2

```
/*! @name SW_MUX_CTL_PAD - SW_MUX_CTL_PAD_GPIO_EMC_00 SW MUX Control Register..SW_MUX_CTL_PAD_GPIO_SD_B1_11 SW MUX Control Register */
/*! @{ */
#define IOMUXC_SW_MUX_CTL_PAD_MUX_MODE_MASK      (0x7U)
#define IOMUXC_SW_MUX_CTL_PAD_MUX_MODE_SHIFT     (0U)
/*! MUX_MODE - MUX Mode Select Field.
 *  0b000..Select mux mode: ALT0 mux port: SEMC_DATA00 of instance: semc
 *  0b001..Select mux mode: ALT1 mux port: FLEXPWM4_PWMA00 of instance: flexpwm4
 *  0b010..Select mux mode: ALT2 mux port: LPSPI2_SCK of instance: lpspi2
 *  0b011..Select mux mode: ALT3 mux port: XBAR1_XBAR_IN02 of instance: xbar1
 *  0b100..Select mux mode: ALT4 mux port: FLEXIO1_FLEXIO00 of instance: flexio1
 *  0b101..Select mux mode: ALT5 mux port: GPIO4_IO00 of instance: gpio4
 */
#define IOMUXC_SW_MUX_CTL_PAD_MUX_MODE(x)        (((uint32_t)(((uint32_t)(x)) << IOMUXC_SW_MUX_CTL_PAD_MUX_MODE_SHIFT)) & IOMUXC_SW_MUX_CTL_PAD_MUX_MODE_MASK)
#define IOMUXC_SW_MUX_CTL_PAD_SION_MASK          (0x10U)
#define IOMUXC_SW_MUX_CTL_PAD_SION_SHIFT         (4U)
/*! SION - Software Input On Field.
 *  0b1..Force input path of pad GPIO_EMC_00
 *  0b0..Input Path is determined by functionality
 */
#define IOMUXC_SW_MUX_CTL_PAD_SION(x)            (((uint32_t)(((uint32_t)(x)) << IOMUXC_SW_MUX_CTL_PAD_SION_SHIFT)) & IOMUXC_SW_MUX_CTL_PAD_SION_MASK)
/*! @} */
```

Figure 42 – NXP MIMXRT1052.h

Figure 43 – 2023.03.16 [400-6] [UNREDACTED] Exhibit G-2

```
#define PCA9535_INPUT_PORT0     0x00
#define PCA9535_INPUT_PORT1     0x01

#define PCA9535_OUTPUT_PORT0    0x02
#define PCA9535_OUTPUT_PORT1    0x03

#define PCA9535_POLARITY_PORT0  0x04
#define PCA9535_POLARITY_PORT1  0x05

#define PCA9535_CONF_PORT0      0x06
#define PCA9535_CONF_PORT1      0x07
```

Figure 44 – NXP pca9535.c

91. Mr. Crozier also appears to be making an assumption that the BSP work referenced in the September 2022 email conversations is related to the SRTOS. But I have reviewed the Skyryse source code and confirmed that any content that was in the SRTOS filepaths that Mr. Crozier appears to be relying on for his opinion was removed and is no longer there. To the extent Skyryse is developing a BSP, it appears to be based on unrelated code.

H. Arduino Files

92. Mr. Pixley describes in his declaration that he "found that on February 6, and February 9, 2022, [Mr. Dao] copied 39,278 files to an external USB drive," and that "[a]pproximately one week later on February 15, 2021, Tri Dao plugged the same external USB drive into his Skyryse laptop (IDS S0022) and copied 7,679 files (of the 39,279 file) he originally copied from his Moog laptop to his Skyryse laptop. I understand that Mr. Pixley and/or Mr. Crozier had access to those files allegedly transferred by Mr. Dao to his Skyryse laptop. [redacted]. However, Neither Mr. Pixley nor Mr. Crozier provide any opinion regarding the contents of the files.

93. I have reviewed the files that Mr. Dao allegedly transferred to his Skyryse laptop and can confirm that these files relate to "Arduino," which is an open-source hardware and software company, project, and user community that

designs and manufactures single-board microcontrollers and microcontroller kits for building digital devices.

94. For example, a significant portion of the files stored in a [redacted] in the iDS container [redacted]. can be found at the single public repository of example source code *Arduino core for ESP8266 WiFi chip*.[54] There are also many files, such as *..\temp\MyArduino\My_Tutorials\generated_examples\Blink\Blink.ino*, that can be found in the Arduino tutorials themselves like the *Liquid Crystal Displays (LCD) with Arduino* tutorial.[55] Still other files, such as *..\temp\MyArduino\My_Tutorials\My_Program\Morse\Morse.cpp* can be found shared in or incorporating contents from discussions in the Arduino forum like, *Extension to Morse Library example*.[56] Similarly, the contents of other source code files can be found in other publicly available sources, such as *ChibiOS-Arduino*.[57] Based on my comparison of the with this publicly available information, I have found that the files Mr. Dao allegedly transferred to his Skyryse laptop relate to "Arduino" which is not Moog's.

VI. CONCLUSION

95. I found that documents and source code that Mr. Crozier and Mr. Pixley point to as purported Moog non-public information includes substantial information from the public domain as well as information that was developed by Mr. Pilkington before his employment began at Moog. Mr. Crozier and Mr. Pixley

---

[54] "Arduino core for ESP8266 WiFi chip" Github, Inc. WEB. https://github.com/esp8266/Arduino. Accessed on April 21, 2023.
[55] "Liquid Crystal Displays (LCD) with Arduino" Arduino. WEB. https://docs.arduino.cc/learn/electronics/lcd-displays. Accessed on April 21, 2023.
[56] "Extension to Morse Library example" Arduino. WEB. https://forum.arduino.cc/t/extension-to-morse-library-example/699472. Accessed on April 21, 2023.
[57] "ChibiOS-Arduino" Github, Inc. WEB. https://github.com/greiman/ChibiOS-Arduino/blob/master/libraries/ChibiOS_AVR/examples/chSemaphore/chSemaphore.ino. Accessed on April 21, 2023.

have failed to provide facts suggesting that the information they claim to be Moog's non-public information is actually Moog's and was not originated or derived from other sources, and is not publicly available.

96. I declare under penalty of perjury that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true.

97. I understand that additional materials may be produced. I therefore reserve the right to supplement or amend my opinion, as expressed in this Declaration, following the production of additional materials and further analysis of the current or additional materials.

Executed on April 24, 2023, in Mountain View, CA.

Nikolaus Baer